Robots at the Edge of the Cloud

Rupak Majumdar(✉)

Max Planck Institute for Software Systems (MPI-SWS),
Kaiserslautern, Saarbrücken, Germany
rupak@mpi-sws.org

Abstract. Computers have come a long way from their roots as fast calculating devices. We live in a world in which computers collect, store, and analyze huge volumes of data. We are seeing the beginnings of a new revolution in the use of computers. In addition to collecting and analyzing data, computers are influencing the physical world and interacting autonomously, and in complex ways, with large groups of humans. These *cyber-physical-social* systems have the potential to dramatically alter the way we lead our lives. However, designing these systems in a reliable way is a difficult problem. In this paper, we enumerate a set of research challenges that have to be overcome in order to realize the potential of cyber-physical-social systems.

1 Motivation

The computer has come a long way from its initial role as a fast calculating device. We live in a world where a large number of geographically distributed and physically embedded computing devices increasingly participate in our everyday actions. Our digital activities generate and consume data at unprecendented volumes. This data is collected, stored, and combined in novel ways. Storage, communication, and processing of this data has moved out of individual workstations into large ensembles of geographically distributed computers ("cloud computers") connected via the Internet and dynamically managed for data processing tasks. Data processing in the cloud has revolutionized the way we approach large-scale design and deployment of software systems. Over the past decade, a similar revolution is happening in the monitoring of the physical world through large swarms of sensors wirelessly connected with each other and with a cloud computing backbone (called variously "Internet of Things" (IoT) or "sensory swarm" [31]).

The next wave in this progression is in large-scale interaction with the physical world through autonomous systems with actuation capabilities and symbiotic relationships of these autonomous systems with large groups of humans. These autonomous systems, call them *robots*, will connect wirelessly with each other as well as with the cloud and the sensor swarm. They will interact with large groups of people, and their actuation capabilities will allow them to modify the state of the physical world. For the purposes of this paper, and keeping with marketing practice, let us call this next progression the "Internet of Robots" (IoR).

M. Chechik and J.-F. Raskin (Eds.): TACAS 2016, LNCS 9636, pp. 3–13, 2016.
DOI: 10.1007/978-3-662-49674-9_1

The notion of "closing the control loop" on sensor networks is not novel (see, e.g., [44]), and indeed, the whole field of cyber-physical systems studies the interaction between software and the physical world. However, the potential for IoR today, with the computing and sensing infrastructure available through the cloud and IoT, is much greater. On the one hand, we can expect systems with a large number of dynamically interacting autonomous agents co-ordinating through the cloud. On the other hand, we can expect systems in which software agents and humans co-operate towards a common goal. In contrast to the "traditional" view where the human is in charge and the machines perform his or her bidding, the IoR vision is that human agents and software or robotic agents interact equally, or even with robots in charge.

While the IoR vision holds enormous promise, as with other grand visions, realizing it requires overcoming a number of very hard research problems. In this paper, we discuss some challenges for IoR, and posit that implementing the IoR vision is a grand challenge for computer science. We structure the problems in three core directions: challenges in the correct design of core algorithmic components (controller synthesis), challenges in software engineering, and challenges in human-robot interaction.

2 Formal Design of Control Systems

At the core of IoR is the notion of *feedback control* of mixed discrete-continuous dynamical systems. Feedback control has a long history in both the continuous world of dynamical systems and in the discrete world of automata theory.

In continuous control, one starts with a model of the system in continuous time, where the continuous state of the system evolves based on the current state, a control input, and a disturbance input. The goal is to provide *feedback* to the system through the control input, that depends on the sensed value of the state, so that the controlled system has "good" properties. Typically, the properties studied in control are stability or performance. When restricted to these properties, under suitable restrictions on the dynamics, methods from continuous control theory show how to synthesize the feedback controller.

In control of discrete systems, usually called *reactive synthesis*, one models the system as a two-player game on graphs, and the goal is to come up with a strategy, a state machine that looks at the history of the game and defines the next move in the game, so that the outcome satisfies "good" properties. Properties are typically specified in a temporal logic such as linear-time temporal logic (LTL) or using automata [17].

The combination of continuous and discrete dynamics leads to *cyber-physical* systems (CPS). In a cyber-physical system, the discrete component ranges over nodes of a graph —as in reactive synthesis— and for each node of the graph, there is a separate continuous dynamics. Cyber-physical systems arise in control problems when one mixes the higher-level logical decision making (e.g., planning a trajectory to reach a goal while avoiding obstacles) with lower-level dynamics (e.g., navigation). At the level of the continuous dynamics, the specification is, as before, related to stability and performance. At the level of the discrete state, the specification is, as in reactive synthesis, given as a temporal logic formula.

The controller for a cyber-physical system combines a strategy at the discrete level with strategies at the continuous level. Traditionally, such controllers would be designed "by hand" and verified through extensive simulation or through symbolic techniques such as model checking. Recent research attempts to synthesize controllers directly from the specification and the model [30,34,46]. Typically, these synthesis techniques compute a finite-state abstraction of the continuous dynamical system and apply reactive synthesis to the abstraction. One step of the discrete strategy —for example, a request to go from one abstract state to the next— can be refined to a continuous controller in the original continuous system. Under certain assumptions on the dynamics, one can show that the original and the abstracted systems are related by an ε-bisimulation relation [24,46]. This guarantees that controllers synthesized on the abstraction can be implemented on the original system.

While initial results on controller synthesis through abstractions is encouraging, there are several difficult technical challenges before the techniques can be applied more widely and to larger classes of systems. We outline some key research questions.

Scalability of Synthesis. The major challenge in using formal synthesis techniques is their scalability. The abstraction of a continuous system yields a discrete system which is exponential in the dimension. In addition, reactive synthesis algorithms, even with symbolic implementations, are expensive (cubic for broad classes of properties [13], but doubly exponential for full linear temporal logic [37]). When the modeling paradigm is extended to include probabilities or other numerical parameters, the problem is even harder.

Recent approaches attempt to get around the scalability by adapting receding horizon control techniques for temporal logic [52]. There are also some interesting initial approaches based on deductive approaches [20], compositional synthesis [38,41] and hierarchical decompositions [43]. In addition to techniques for state-space reduction, an important open direction is to handle dynamically changing specifications in open environments.

Notions of Robustness. A system is *robust* if small changes to its inputs cause small changes in its outputs. Robustness is a classical notion in control theory and a natural requirement when designing control systems. However, appropriate notions of robustness are difficult to obtain for cyber-physical systems. Physical systems are modeled and analyzed using continuous mathematics and concepts such as continuity are readily available to help describe robustness. On the other hand, discrete systems are modeled and analyzed using discrete mathematics for which it is far less obvious what a meaningful notion of robustness can be. There are several current attempts to define notions of robustness [11,47]. However, a challenge is to come up with a definition that is broadly applicable and has good algorithmic properties.

Quantitative Properties. Related to robustness are *quantitative* specification languages for synthesis. A specification in LTL classifies system behaviors as "true" or "false." In many cases, such specifications are too strict, and it is preferable to use a quantitative formalism that associates a numerical score with system behaviors. Recent work in the theory of quantitative languages and synthesis [12,14,16], in quantitative logics such as signal temporal logic (STL) [21], and in metrics on systems such as the Skorokhod metric [19,35], move in this direction. The use of quantitative specifications also opens the door to more data-driven approaches that optimize or learn system parameters [23,51]. Combinations of learning with synthesis is an interesting emerging area.

System Co-design. The controller is one component of a complex stack integrating sensing, computation, communication, and actuation running several control loops at various different levels of granularity. An end-to-end design, which co-designs the controller along with other components of the system can achieve better resource usage than one which designs each component in isolation. For example, by designing a controller robust to intermittent steps in which the control input is not computed, one can schedule more processes in the same processor without sacrificing control performance [36,42,45]. At the same time, co-design techniques may involve loss of modularity in the design. It is a challenging question whether tradeoffs between design choices of different components can be captured in an abstract interface —controller-scheduler co-design is one example where this is possible [42].

Co-design also requires reasoning about the underlying architecture. For example, in order to guarantee a certain system performance, it may be necessary to provide bounds on worst case execution times or worst case latencies. This is a hard problem, and may require a fundamental redesign of architectural elements for cyber-physical systems [7,33].

Co-design considers the various algorithmic components of a system together. A new set of challenges arise when we consider the software implementation of a system, which we describe next.

3 Programming Model and the Software Stack

Formal synthesis and verification is a key step towards more reliable and large-scale IoR systems. However, synthesis of controllers is only the "core" algorithm. For end-to-end development, these algorithms must be embedded in a software stack. We now discuss the challenges of developing programming models and software infrastructure for IoR applications.

We start with an analogy in the cloud computing scenario. Cloud computing abstracts the computing, distribution, communication, and storage needs of a large-scale, distributed application. The end user can write computational tasks focusing on the functionality. The cloud infrastructure manages physical resources for the computation such as compute-servers, distribution, and fault tolerance. The cloud can dynamically provision additional resources for computation and storage, or distribute or replicate a data structure across geographically separated infrastructure. However, for the most part, this is transparent to the user application.

A dominant application on the cloud is statistical analysis of large data sets. For this application, there is a declarative programming model (querying data in specialized languages such as Pig Latin [5] or HiveQL [4]) that compiles into a computational model (map-reduce [3,18] or Spark [6]) for fast execution on parallel machines. The programmer's view of the data is abstracted to centralized database tables, and the query is made at a logical level. The infrastructure takes care of executing the "program" on distributed and dynamic infrastructure. This includes not only scheduling parallel jobs for low latency but also fault tolerance and replication. While not a panacea for all applications, the abstraction enables the separation between the programming abstraction (logical operations on a dataset) and the infrastructure on which the operations are performed.

We lack a corresponding "programming model" for large-scale IoR applications. Currently, applications are written in low level programming languages and use ad hoc mechanisms to implement layering between logical task models and the underlying continuous controllers. Communication is mediated through middleware such as ROS [39], but the resulting message-passing programs are difficult to design and verify, especially when the number of components grow large and when components can dynamically enter or leave the system.

A key design challenge for IoR is to develop programming tools for cyber-physical systems. We outline some research challenges in this direction.

Programming Models and Run-time Systems. One major success of programming models we use today is that they abstract from the real world. The "step" of a Turing machine is a purely logical step. When we program, we do not, for most applications, reason about the details of the physical world. This abstraction breaks when we design cyber-physical systems where the controller must react to events within a given (real) time bound. Indeed, dealing with the real world and real time is one reason designing and verifying these systems is so difficult. The difficulty is compounded when we consider large ensembles of autonomous agents concurrently sensing and actuating the physical world. It is yet unclear what programming models will enable ordinary developers design large-scale IoR applications. One possibility is the programming idiom of actors [1], extended to faithfully represent interactions in the physical world. A different possibility is a specialized programming model such as the globally asynchronous locally synchronous (GALS) model that abstracts out the part of the timing behavior that the underlying compiler and run time systems enforce [9,15,49]. Very likely, declarative techniques to specify controller behaviors —e.g., in LTL or STL— will be integrated within the programming model; the compiler will be expected to generate the code that enforces these behaviors at run time.

Developing languages, compilers, and run-time systems for large-scale sensing, co-ordination, computing, and actuation is an outstanding open problem.

Managing Uncertainty. A second challenge is to incorporate uncertainty as a first-class construct in the language. IoR applications will necessarily work in environments that are not completely specified, or whose behaviors may change over time in unexpected ways. There is a lot of recent work on introducing uncertainty and probabilistic reasoning in programming languages [26,27]. Integration of uncertainty management with controller synthesis in a programming model is likely to be the next step.

Managing Dynamic Resources. Embedded control systems are currently programmed with well-defined resource requirements at compile time. Since their correctness depends on real-time requirements, a conservative static analysis bounds required resources and pre-allocates these resources. Such a programming model can be overly pessimistic in dynamic environments where requirements change and resources can be provisioned dynamically. It is a challenge to set up a programming model and run-time where critical resources are statically allocated (to ensure basic safety) but other resources can be dynamically provisioned.

4 Cyber-Physical-Social Systems

In many IoR applications, groups of autonomous agents interact with humans. For example, in an autonomous vehicle, the human may be kept in the loop to compensate for driving conditions not familiar to the autonomous driver. In a traffic management scenario involving both autonomous and human drivers, a central server may provide route suggestions to ensure optimal flow of traffic based on dynamically collected data, while individual drivers —human and autonomous— may decide to follow the suggestions or not, based on individual rational preferences. In an energy distribution scenario, human-operated electric vehicles may be used to store energy and redistribute it in the grid. In these and many other emerging scenarios, humans interact closely with computers and controllers in a dynamic fashion [29]. These systems raise a number of new challenges in design, implementation, and analysis.

First, the interaction between humans and computers may not be "one way" —the human initiating a task that the controller implements— but involve cooperation between the two, and even be initiated by the controller. Second, formal reasoning about such systems requires understanding social behavior and the incentives that enable human participants to act in a way that optimizes the overall system behavior. Third, these systems must implement infrastructure to manage privacy, accountability, compliance, and reputation. We focus on two directions.

Specification Challenges. Computers can follow algorithms that enforce logically-specified behaviors. Unfortunately, it is hard to enforce similar logical specifictions for human behaviors. Instead, humans participate in activities motivated by incentives, such as the desire for a particular beneficial outcome, or extrinsic motivations such as money or reputation. Thus, the design of cyber-social systems requires not only logical specifications but also incentive mechanisms that ensure the participants engage in behaviors that are beneficial to the system. There is some initial work on the modeling of human participants in human-in-the-loop control (see, e.g., [22,32]), but a unified science for formal design of cyber-physical-social systems remains a big challenge. Traditionally, design of protocols for rational agents is the realm of (algorithmic) mechanism design. An open question is whether approaches to synthesis from control, reactive synthesis, and mechanism design can be combined profitably.

At the programming level, human-agent collectives also introduce new challenges. First, the natural interaction of humans with robots is not at the level of code or of logical specifications but at the level of natural languages or gestural user interfaces. Second, the programming model has to support incentive mechanisms that allow groups of humans to come together for a system task, in cooperation with autonomous participants. Designing such programming abstractions that interoperate between code, natural user interfaces, and incentive mechanisms is a hard problem.

We are encouraged by some programming abstractions that are emerging in projects such as participatory sensing [40,48] and crowd-sourced computation (such as the Mechanical Turk API [2] and its embedding in programming platforms [10,50]). It remains to be seen how such platforms can be integrated with control and co-ordination capabilities.

Privacy, Accountability, and Trust. A system that allows large scale interaction between humans and autonomous agents also leads to social and ethical problems. Human social actions often follow incentives such as social responsibility and reputation: we behave in the way we do because we care about how our actions are perceived by others, and we are held accountable for our interactions with other humans. It is unclear how norms of social behavior change when we interact with robots. Can we "hurt the feelings" of an autonomous controller by ignoring its suggestion? Moreover, when things go wrong, due to errors or malicious behavior, how is accountability shared between human participants and autonomous ones? Many of these questions involve social sciences or law in addition to engineering and computer science.

Related to the problem of accountability is the problem of privacy. The use of information relating to individuals may be necessary to engineer a system to its optimal outcomes, but revealing personal information may not be allowed due to individual preferences or regulatory limitations. The tradeoff between accountability and privacy, or related tradeoffs between privacy and trust, must be understood as we design more complex systems [8]. This problem is already relevant in social computing systems without physical controllers; it takes a larger role in complex applications where robots actuate the physical world.

5 Conclusion

IoR applications have the potential to transform the way we interact with computers and with each other. The road to reliable and massive-scale IoR applications is long, and has many exciting research challenges, both technical and social. While the list of problems in this paper is partial, they already demonstrate the richness of the research landscape.

Acknowledgements.. This work is partially funded by the ERC Synergy grant ImPACT. Thanks to Dmitry Chistikov, Samira Farahani, and Anne-Kathrin Schmuck for useful discussions on these topics. I was inspired by several excellent overview articles on the topic of this paper [28,29,31].

References

1. Agha, G.A.: ACTORS - A Model of Concurrent Computation in Distributed Systems. MIT Press series in artificial intelligence. MIT Press, Cambridge (1990)
2. Amazon mechanical turk. https://aws.amazon.com/documentation/mturk/
3. Apache hadoop. https://hadoop.apache.org/
4. Apache hive. https://hive.apache.org/
5. Apache pig. https://pig.apache.org/
6. Apache spark. https://spark.apache.org/
7. Axer, P., Ernst, R., Falk, H., Girault, A., Grund, D., Guan, N., Jonsson, B., Marwedel, P., Reineke, J., Rochange, C., Sebastian, M., von Hanxleden, R., Wilhelm, R., Yi, W.: Building timing predictable embedded systems. ACM Trans. Embedded Comput. Syst. **13**(4), 82:1–82:37 (2014)
8. Backes, M., Druschel, P., Majumdar, R., Weikum, G.: Impact: privacy, accountability, compliance, and trust in tomorrow's Internet. ERC Synergy Grant White Paper (2015). http://www.impact-erc.eu/src/doc/imPACT-whitepaper.pdf
9. Balarin, F., Chiodo, M., Giusto, P., Hsieh, H., Jurecska, A., Lavagno, L., Passerone, C., Sangiovanni-Vincentelli, A., Sentovich, E., Suzuki, K., Tabbara, B.: The Polis Approach. The Springer International Series in Engineering and Computer Science, vol. 404. Kluwer Academic, Dordrecht (1997)
10. Bernstein, M.S., Chi, E.H., Chilton, L.B., Hartmann, B., Kittur, A., Miller, R.C.: Crowdsourcing, human computation: systems, studies and platforms. In: Tan, D.S., Amershi, S., Begole, B., Kellogg, W.A., Tungare, M. (eds.) Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Extended Abstracts Volume, Vancouver, BC, Canada, 7–12 May, 2011, pp. 53–56. ACM (2011)
11. Bloem, R., Chatterjee, K., Greimel, K., Henzinger, T.A., Hofferek, G., Jobstmann, B., Könighofer, B., Könighofer, R.: Synthesizing robust systems. Acta Inf. **51**(3–4), 193–220 (2014)
12. Bloem, R., Chatterjee, K., Henzinger, T.A., Jobstmann, B.: Better quality in synthesis through quantitative objectives. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 140–156. Springer, Heidelberg (2009)
13. Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., Sa'ar, Y.: Synthesis of reactive(1) designs. J. Comput. Syst. Sci. **78**(3), 911–938 (2012)
14. Cerný, P., Henzinger, T.A., Radhakrishna, A.: Simulation distances. Theor. Comput. Sci. **413**(1), 21–35 (2012)
15. Chapiro, D.M.: Globally asynchronous locally synchronous systems. PhD thesis (1984)
16. Chatterjee, K., Doyen, L., Henzinger, T.A.: Quantitative languages. ACM Trans. Comput. Log. **11**(4), 23 (2010)

17. Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. MIT Press, Cambridge (1999)
18. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
19. Deshmukh, J.V., Majumdar, R., Prabhu, V.S.: Quantifying conformance using the skorokhod metric. In: Kroening, D., Păsăreanu, C.S. (eds.) CAV 2015. LNCS, vol. 9207, pp. 234–250. Springer, Heidelberg (2015)
20. Dimitrova, R., Majumdar, R.: Deductive control synthesis for alternating-time logics. In: Mitra, T., Reineke, J. (eds.) 2014 International Conference on Embedded Software, EMSOFT 2014, New Delhi, India, 12–17 October, 2014, pp. 14:1–14:10. ACM (2014)
21. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: Chatterjee, K., Henzinger, T.A. (eds.) FORMATS 2010. LNCS, vol. 6246, pp. 92–106. Springer, Heidelberg (2010)
22. Feng, L., Wiltsche, C., Humphrey, L., Topcu, U.: Controller synthesis for autonomous systems interacting with human operators. In: Bayen, A.M., Branicky, M.S. (eds.) Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems, Seattle, WA, USA, 14–16 April, pp. 70–79. ACM (2015)
23. Fu, J., Topcu, U.: Probably approximately correct MDP learning, control with temporal logic constraints. In: Fox, D., Kavraki, L.E., Kurniawati, H. (eds.) Robotics: Science and Systems X, University of California, Berkeley, USA, 12–16 July, 2014 (2014)
24. Girard, A., Pappas, G.J.: Approximate bisimulation: a bridge between computer science and control theory. Eur. J. Control **17**(5–6), 568–578 (2011)
25. Girard, A., Sankaranarayanan, S. (eds.) Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, HSCC 2015, Seattle, WA, USA, 14–16 April, 2015. ACM (2015)
26. Gordon, A.D., Aizatulin, M., Borgström, J., Claret, G., Graepel, T., Nori, A.V., Rajamani, S.K., Russo, C.V.: A model-learner pattern for bayesian reasoning. In: Giacobazzi, R., Cousot, R. (eds.) The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2013, Rome, Italy - 23–25 January, pp. 403–416. ACM (2013)
27. Gordon, A.D., Henzinger, T.A., Nori, A.V., Rajamani, S.K.: Probabilistic programming. In: Herbsleb, J.D., Dwyer, M.B. (eds.) Proceedings of the on Future of Software Engineering, FOSE 2014, Hyderabad, India, 31 May - 7 June, 2014, pp. 167–181. ACM (2014)
28. Henzinger, T.A., Sifakis, J.: The discipline of embedded systems design. IEEE Comput. **40**(10), 32–40 (2007)
29. Jennings, N.R., Moreau, L., Nicholson, D., Ramchurn, S.D., Roberts, S.J., Rodden, T., Rogers, A.: Human-agent collectives. Commun. ACM **57**(12), 80–88 (2014)
30. Kress-Gazit, H., Fainekos, G.E., Pappas, G.J.: Temporal-logic-based reactive mission and motion planning. IEEE Trans. Robot. **25**(6), 1370–1381 (2009)
31. Lee, E.A., Hartmann, B., Kubiatowicz, J., Rosing, T.S., Wawrzynek, J., Wessel, D., Rabaey, J.M., Pister, K., Sangiovanni-Vincentelli, A.L., Seshia, S.A., Blaauw, D., Dutta, P., Fu, K., Guestrin, C., Taskar, B., Jafari, R., Jones, D.L., Kumar, V., Mangharam, R., Pappas, G.J., Murray, R.M., Rowe, A.: The swarm at the edge of the cloud. IEEE Des. Test **31**(3), 8–20 (2014)
32. Li, W., Sadigh, D., Sastry, S.S., Seshia, S.A.: Synthesis for human-in-the-loop control systems. In: Ábrahám, E., Havelund, K. (eds.) TACAS 2014 (ETAPS). LNCS, vol. 8413, pp. 470–484. Springer, Heidelberg (2014)

33. Liu, I., Reineke, J., Broman, D., Zimmer, M., Lee, E.A.: A PRET microarchitecture implementation with repeatable timing and competitive performance. In: 30th International IEEE Conference on Computer Design, ICCD 2012, Montreal, QC, Canada, 30 September- 3 October, 2012, pp. 87–93. IEEE Computer Society (2012)
34. Liu, J., Ozay, N., Topcu, U., Murray, R.M.: Synthesis of reactive switching protocols from temporal logic specifications. IEEE Trans. Automat. Contr. **58**(7), 1771–1785 (2013)
35. Majumdar, R., Prabhu, V.S.: Computing the skorokhod distance between polygonal traces. In: Girard, A., Sankaranarayanan, S. (eds.) [25], pp. 199–208
36. Majumdar, R., Saha, I., Zamani, M.: Synthesis of minimal-error control software. In: Jerraya, A., Carloni, L.P., Maraninchi, F., Regehr, J. (eds.) Proceedings of the 12th International Conference on Embedded Software, EMSOFT 2012, part of the Eighth Embedded Systems Week, ESWeek 2012, Tampere, Finland, 7–12 October, 2012, pp. 123–132. ACM (2012)
37. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: Conference Record of the Sixteenth Annual ACM Symposium on Principles of Programming Languages, Austin, Texas, USA, 11–13 January, pp. 179–190. ACM (1989)
38. Pola, G., Pepe, P., Di Benedetto, M.D.: Compositional symbolic models for networks of incrementally stable control systems. CoRR, abs/1404.0048 (2015)
39. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Andrew, Y. Ng. ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
40. Reddy, S., Mun, M.Y., Burke, J., Estrin, D., Hansen, M.H., Srivastava, M.B.: Using mobile phones to determine transportation modes. TOSN **6**(2), 23 (2010)
41. Rungger, M., Zamani, M.: Compositional construction of approximate abstractions. In: Girard, A., Sankaranarayanan, S. (eds.) [25], pp. 68–77
42. Saha, I., Baruah, S., Majumdar, R.: Dynamic scheduling for networked control systems. In: Girard, A., Sankaranarayanan, S. (eds.) [25], pp. 98–107
43. Schmuck, A.-K., Majumdar, R.: Dynamic hierarchical reactive controller synthesis. CoRR, abs/1510.07246 (2015)
44. Sinopoli, B., Sharp, C., Schenato, L., Schaffert, S., Sastry, S.: Distributed control applications within sensor networks. IEEE Proc. **91**(8), 1235–1246 (2003)
45. Soudbakhsh, D., Phan, L.T.X., Sokolsky, O., Lee, I., Annaswamy, A.: Co-design of control and platform with dropped signals. In: Lu, C., Kumar, P.R., Stoleru, R. (eds.) ACM/IEEE 4th International Conference on Cyber-Physical Systems (with CPS Week), ICCPS 2013, Philadelphia, PA, USA, 8–11 April, 2013, pp. 129–140. ACM (2013)
46. Tabuada, P.: Verification and Control of Hybrid Systems - A Symbolic Approach. Springer, Heidelberg (2009)
47. Tabuada, P., Caliskan, S.Y., Rungger, M., Majumdar, R.: Towards robustness for cyber-physical systems. IEEE Trans. Automat. Contr. **59**(12), 3151–3163 (2014)
48. Tangmunarunkit, H., Hsieh, C.-K., Longstaff, B., Nolen, S., Jenkins Ketcham, J., Ketcham, C., Selsky, J., Alquaddoomi, F., George, D., Kang, J., Khalapyan, Z., Ooms, J., Ramanathan, N., Estrin, D.: Ohmage: a general and extensible end-to-end participatory sensing platform. ACM TIST **6**(3), 38 (2015)
49. Tripakis, S., Pinello, C., Benveniste, A., Sangiovanni-Vincentelli, A.L., Caspi, P., Di Natale, M.: Implementing synchronous models on loosely time triggered architectures. IEEE Trans. Comput. **57**(10), 1300–1314 (2008)
50. Trushkowsky, B., Kraska, T., Sarkar, P.: Answering enumeration queries with the crowd. Commun. ACM **59**(1), 118–127 (2016)

51. Wen, M., Ehlers, R., Topcu, U.: Correct-by-synthesis reinforcement learning with temporal logic constraints. CoRR, abs/1503.01793 (2015)
52. Wongpiromsarn, T., Topcu, U., Murray, R.M.: Receding horizon control for temporal logic specifications. In: Johansson, K.H., Yi, W. (eds.) Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2010, Stockholm, Sweden, 12–15 April, 2010, pp. 101–110. ACM (2010)