Technical Communications of the International Conference on Logic Programming, 2010 (Edinburgh), pp. 289–293
http://www.floc-conference.org/ICLP-home.html

LOGIC PROGRAMMING FOUNDATIONS OF CYBER-PHYSICAL SYSTEMS

NEDA SAEEDLOEI [1]

[1] Department of Computer Science
University of Texas at Dallas
Richardson, TX 75080, USA
E-mail address: neda.saeedloei@student.utdallas.edu

Abstract. Cyber-physical systems (CPS) are becoming ubiquitous. Almost every device today has a controller that reads inputs through sensors, does some processing and then performs actions through actuators. These controllers are discrete digital systems whose inputs are continuous physical quantities and whose outputs control physical (analog) devices. Thus, CPS involve both digital and analog data. In addition, CPS are assumed to run forever, and many CPS may run concurrently with each other. we will develop techniques for faithfully and elegantly modeling CPS. Our approach is based on using *constraint logic programming over reals, co-induction*, and *coroutining*.

1. Introduction and Problem Description

Cyber-physical systems (CPS) are becoming ubiquitous. Almost every device today has a controller that reads inputs through sensors, does some processing and then performs actions through actuators. Examples include controller systems in cars (Anti-lock Brake System, Cruise Controllers, Collision Avoidance, etc.), automated manufacturing, smart homes, robots, etc. These controllers are discrete digital systems whose inputs are continuous physical quantities (e.g., time, distance, acceleration, temperature, etc.) and whose outputs control physical (analog) devices. Thus, CPS involve both digital and analog data. In addition, CPS are assumed to run forever, and many CPS may run concurrently with each other [Lee08, Gup06].

CPS have the following four characteristics [Lee08, Gup06]: (i) they perform discrete computations, (ii) they deal with continuous quantities, (iii) they are concurrent, and (iv) they run forever. Due to fundamentally discrete nature of computation, researchers have had difficulty dealing with continuous quantities in computations (typical approaches discretize continuous quantities, e.g., time). Likewise, modeling of perpetual computations is not well understood (only recently, techniques such as co-induction [Sim07, Gup07] have been introduced to formally model rational, infinite computations). Concurrency is reasonably well understood, but when combined with continuous quantities and with perpetual computations, CPS become extremely hard to model faithfully. In this research work we will develop techniques for faithfully and elegantly modeling CPS for which no good formalisms exist within computer science.

Key words and phrases: Cyber-Physical Systems, Constraint Logic Programming over reals, Co-induction, Coroutining.

Technical Communications of the 26th International Conference on Logic Programming, Edinburgh, July, 2010
Editors: Manuel Hermenegildo, Torsten Schaub
LIPIcs - Leibniz International Proceedings in Informatics. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany
Digital Object Identifier: 10.4230/LIPIcs.ICLP.2010.289

2. Background and Overview of the Existing Literature

CPS are highly complex systems for which today's computing and networking technologies do not provide adequate foundations. In fact, Edward Lee states [Lee08]:

> Cyber-physical systems are integrations of computation with physical processes. Embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computation and vice versa. In the physical world, the passage of time is inexorable and concurrency is intrinsic. Neither of these properties is present in today's computing and networking abstractions.

Lee goes on to argue that "the mismatch between these abstractions and properties of physical processes impede technical progress." Thus according to Lee, a major challenge is to find the right abstractions for CPS. Similarly, Rajesh Gupta [Gup06] urges researchers "to achieve the goal of semantic support for location and time at all levels," and address the following technical problems for CPS:

(1) "How do we capture location (and timing) information into CPS models that allows for validation of the logical properties of a program against the constraints imposed by its physical (sensor) interaction."
(2) "What are useful models for capturing faults and disconnections within the coupled physical-computational system? ..."
(3) "What kind of properties that can be verified, ..."
(4) "What programming model is best suited for CPS applications ..."

3. Goal of the Research

The goal of our research is to develop techniques for faithfully and elegantly modeling cyber-physical systems. our approach is based on using logic programming for modeling computations, constraint logic programming for modeling continuous physical quantities, co-induction for modeling perpetual execution and coroutining for modeling concurrency in CPS. CPS are thus represented as coroutined co-inductive constraint logic programs which are subsequently used to elegantly verify cyber-physical properties of the system relating to safety, liveness and utility. This logic program can also be used for automatically generating implementation code for the CPS.

4. Current Status of the Research and Preliminary Results Accomplished

We assume that most CPS are state machines (finite automata) that control physical systems. In our formalism, state machines are modeled as logic programs [Llo87, Ste94], physical quantities are represented as continuous quantities (i.e., not discretized) and the constraints imposed on them by CPS physical interactions are faithfully modeled with *constraint logic programming over reals (CLP(R))* [Jaf94]. By considering co-inductive logic programming [Bar96, Gup07], we are able to model the non-terminating nature of CPS, and finally concurrency will be handled by allowing *coroutining* within logic programming computations.

Hybrid automata (of which timed automata and pushdown timed automata are instances) constitute the foundations for CPS. We have developed a general framework based on constraint logic programming and co-induction for modeling/verifying CPS [Sae10b].

The formalism that are used in this framework are timed automata and pushdown timed automata (PTA) which can be computationally modeled by combination of co-inductive logic programming (or Co-LP) and CLP(R). These can be generalized to hybrid automata and pushdown hybrid automata. We have developed a general method of converting timed automata and PTA to co-inductive CLP(R) programs. The method takes the description of a pushdown timed automaton (timed automaton) and generates a co-inductive constraint logic program over reals. We have shown how a co-inductive CLP(R) rendering of a pushdown timed automaton can be used to verify safety and liveness properties of complex timed systems. We have illustrated the effectiveness of our approach by showing how the well-known *generalized railroad crossing (GRC)* problem [Hei94] can be naturally modeled, and how its various safety and utility properties can be elegantly verified.

We have also developed timed grammars as a simple and natural formalism for describing timed languages. Timed grammars describe words that have real-time constraints placed on the times at which the words' symbols appear. Timed grammars can be generalized to hybrid grammars to model other types of continuous phenomena.

We extended the concept of context-free grammars (CFGs) to timed context-free grammars (TCFGs) and timed context-free ω-grammars (ω-TCFGs for brevity) [Sae10a]. Informally, a timed context-free grammar is obtained by associating clock constraints with terminal and non-terminal symbols appearing in the productions of a CFG. Timed context-free grammars describe timed context-free languages (TCFLs). A TCFL contains those strings that are accepted by the underlying untimed CFG but which also satisfy the timing constraints imposed by the associated clock constraints. Timed context-free ω-grammars describe timed context-free languages containing infinite-sized words, and are a generalization of timed ω-regular languages recognized by *timed automata*.

The words in a timed language consist of a sequence of symbols from the alphabet of the language the grammar accepts paired with the time-stamp indicating the time that symbol was seen. Timed languages are useful for modeling complex real-time, hybrid and cyber-physical systems.

We have shown how DCGs together with CLP(R) and co-induction can be used to develop efficient and elegant parsers for timed grammars. We have developed a system that takes an ω-TCFG and converts it into a DCG augmented with co-induction and CLP(R). The resulting co-inductive constraint logic program acts as a parser for the ω-TCFL recognized by the ω-TCFG. We have applied our general method of converting timed grammars to DCGs to the GRC problem with two tracks and presented simple timed context-free ω-grammar for *controller*, *gate*, and *track* components of this problem. The logic programming rendering of these ω-grammars are also generated by our system.

5. Open Issues and Expected Achievements

Our research group has done significant amount of work over the last few years to model CPS. However, most of it was focused on verifying on properties of systems [Sae10b, Sae10a, Ban10, Gup07]. Also, we were focused on solving the harder problems of logically modeling continuous quantities and perpetual nature of these systems. The concurrency aspect received less attention. As part of my research, I will continue my work on specification and verification of CPS but focus also on concurrency exhibited by CPS as well as generation their implementation from specifications in a provably correct manner. Research will be pursued along the following lines:

Timed π-calculus: I am studying the extension of π-calculus with continuous time. π-calculus [San02] is a well known formalism for modeling concurrency. Theoretically, the π-calculus can model concurrency, message exchange as well as infinite computation (through the infinite replication operator '!'), however, it does not deal with modeling of continuous quantities. I am developing an executable operational semantics of π-calculus in which concurrency is modeled by coroutining in logic programming (realized via *delay declarations of Prolog* [Ste94]) and infinite computations by co-induction [Saeon]. This operational semantics will be extended with continuous real time, which will be modeled with CLP(R). The executable operational semantics thus realized will automatically lead to an implementation of the timed π-calculus. The timed π-calculus will be used to model the GRC more faithfully and to verify its safety and utility properties. There is past work on developing executable operational semantics of the π-calculus (but not timed π-calculus) [Yan], that is based on logic programming, but it falls short as it is unable to model perpetual processes and infinite replication since co-inductive logic programming is a recent concept developed by our group.

Generating Implementation: Thus far we have seen how a cyber-physical system can be specified and its cyber-physical properties verified. We would like to use the specification to also generate the implementation code automatically. This way we can ensure that the implementation is faithful to the (verified) specification.

In order to generate the implementation code, the actions to be taken in the situation that a constraint in not met has to be specified as well. For example considering the GRC problem, what happens if the crossing-gate does not close within 2 units of time since the approach signal of a train was received). That is, normal situations as well as error situations have to be covered. Once the error situations are also specified, then it is relatively straightforward to generate the implementation along with all the exceptions and failsafe checks. Thus, research will be conducted on automatically deriving implementation of CPS from their verified specifications.

Real-life Applications: The modeling and implementation infrastructure we develop will be tested on real-life applications. These applications will come from manufacturing companies.

Acknowledgment

I would like to thank my dissertation advisor, Prof. Gupta, for his constant guidance, support and advice.

References

[Ban10] Ajay Bansal, Neda Saeedloei, and Gopal Gupta. Automated planning under realtime constraints. In *Florida AI Research Symposium*. To appear, 2010.

[Bar96] Jon Barwise and Lawrence Moss. *Vicious circles: on the mathematics of non-wellfounded phenomena*. Center for the Study of Language and Information, Stanford, CA, USA, 1996.

[Gup06] Rajesh Gupta. Programming models and methods for spatiotemporal actions and reasoning in cyber-physical systems. In *NSF Workshop on CPS*. 2006.

[Gup07] Gopal Gupta, Ajay Bansal, Richard Min, Luke Simon, and Ajay Mallya. Coinductive logic programming and its applications. In *ICLP*, pp. 27–44. Springer, 2007.

[Hei94] Constance L. Heitmeyer and Nancy A. Lynch. The generalized railroad crossing: A case study in formal verification of real-time systems. In *IEEE RTSS*, pp. 120–131. 1994.

[Jaf94] Joxan Jaffar and Michael J. Maher. Constraint logic programming: A survey. *J. Log. Program.*, 19/20:503–581, 1994.

[Lee08] Edward A. Lee. Cyber physical systems: Design challenges. In *ISORC*. 2008.

[Llo87] J. W. Lloyd. *Foundations of logic programming / J.W. Lloyd*. Springer-Verlag, Berlin ; New York :, 2nd edn., 1987.

[Sae10a] Neda Saeedloei and Gopal Gupta. Timed definite clause omega-grammars. In *Leibniz International Proceedings in Informatics*. To appear, 2010.

[Sae10b] Neda Saeedloei and Gopal Gupta. Verifying complex continuous real-time systems with coinductive clp(r). In *Languages and Automata Theory*. To appear, 2010.

[Saeon] Neda Saeedloei and Gopal Gupta. Timed pi-calculus and its applications. In preparation.

[San02] Davide Sangiorgi and David Walker. *The pi-Calculus*. Cambridge University Press, 2002.

[Sim07] Luke Simon, Ajay Bansal, Ajay Mallya, and Gopal Gupta. Co-logic programming: Extending logic programming with coinduction. In *ICALP*, pp. 472–483. 2007.

[Ste94] Leon Sterling and Ehud Shapiro. *The art of Prolog (2nd ed.): advanced programming techniques*. MIT Press, Cambridge, MA, USA, 1994.

[Yan] Ping Yang, C. R. Ramakrishnan, and Scott A. Smolka. A logical encoding of the pi-calculus: Model checking mobile processes using tabled resolution. In *VMCAI 2003*, pp. 116–131.