High-level language for IoT and Robotics

Gert Kanter[1], Jüri Vain[1], Elahe Fazeldehkordi[2], and Olaf Owe[3]

[1] Tallinn Technical University, Tallinn, Estonia
[2] Kristiania University College, Oslo, Norway
[3] University of Oslo, Oslo, Norway and University of Agder, Grimstad, Norway
gert.kanter@ttu.ee, juri.vain@ttu.ee,
Elahe.Fazeldehkordi@kristiania.no, olaf@ifi.uio.no

1 Main goal

Our overall goal is to develop a high-level language that unifies concepts from IoT and robotic languages. These concepts include dynamicity, autonomy, as well as safety, security, and privacy of multi-robot missions that act in IoT augmented environments. We will discuss language paradigms and language constructs that are suitable in this setting. We take the executable multi-robot mission specification languages as our starting point and reference from the robotic side. They allow specifying mission goals, navigation and manipulation constraints including timing constraints. From this perspective IoT provides a dynamic knowledge environment for mission completion. OrCcad [9] is one example of such a robotic language designed for real-time control of robot actions. This language as well as other coordination languages for robotics and IoT are based on event systems and state machines. Orchestration of robot systems, for instance as provided by the Resh language [1], is also a relevant perspective to program complex multi-robot applications. The most prominent high-level control systems used for the Robot Operating System (ROS and ROS2) are SMACC [14] (an event-driven, asynchronous, behavioral state machine library for real-time ROS applications written in C++, utilizing the Boost Statechart Library), BehaviorTree.CPP [11] (hierarchical and having more expressive power than traditional FSMs) and SMACH [13] (hierarchical state machines). There might be others with more narrow focus but they are currently not widely adopted by the robotics community. Ada [12] and SPARK [10] are available for ROS(/2), and SPARK was used in [15].

We are interested in a high-level modeling language useful for integration of IoT, robots, and distributed system designs, and supported by modular specification and verification techniques. In particular we will look at the active object paradigm because it comes with a built-in support of autonomous objects, has proved relevant for modeling of IoT systems [7], and supports compositional verification. We are aiming at a language that, on the one hand, is expressive enough to unequivocally specify necessary aspects of multi-robot mission orchestration, and on the other hand, has a simple and compositional semantics to support efficient verification and provability. This combination represents a challenge in the current robotics research landscape. We will focus on high-level, object-oriented modelling of robotic and IoT systems. The language should be executable and allow straight forward translation into more low-level languages. This will allow prototyping and testing of components at an early program development state, and combining them with assurance measures such as model checking and deductive verification.

2 Introduction

Provability of IoT and robotic systems depends on the language used to model the system, its semantics, the kind of system properties to prove, and the techniques used to verify them. One may take a bottom/up approach, starting with a low-level language or one may take a top-down approach, starting with a more abstract language with an expressiveness suitable for IoT/robotic systems. Such a language can be used to model IoT and robotic systems, and if the language is executable, it can be used for simulation and prototyping of IoT systems. From a practical engineering viewpoint, such a high-level language being imperative with standard mechanisms such as object-orientation, IoT and robotic models made in the language can easily be translated to more low-level languages that are natively used in the domain. The second approach also has the advantage that one can define the language and its semantics so that it is amenable to semantic analysis and verification. Our work follow the second approach.

The active object paradigm (AOP) provides a natural way of modelling service-oriented computations, in general, and IoT systems in particular, letting one IoT device correspond to one (or more) active objects. AOP covers the fundamental aspects of IoT systems, such as distribution of concurrent units communicating by message passing, where each unit can run on a device with limited processing power and limited storage. This paradigm supports asynchronous as well as synchronous communication. Each active object is autonomous with its own processor and process queue, and with local scheduling control of the queue [3]. The notion of concurrent object groups, which allow several active objects to be seen as a single object with its own interface [5], allows modelling of robot systems composed of several concurrent objects, be it a robot with internal concurrent parts or a robot swarm with many robots and possibly IoT devices. Since active objects may represent IoT devices, extending this paradigm to robot agents and their functional components seems a natural option.

A number of different language combinations based on the active object paradigm are explored in [7], and in particular the setting with cooperative scheduling through suspension but without the mechanism of futures. The latter is identified as the most attractive one for IoT systems. The analysis of robotic languages shows that the same core concepts are relevant also for programming multi-robot applications, and the constructs for abstract mission description can be built on top of them. The semantic of this language paradigm is mathematically simple, avoiding issues such as aliasing problems, heap management, and the synchronization problems with shared variable systems. A lightweight operating system suffices as a computation platform since process control is built into the language itself. As the hardware of IoT devices and robots range from units with very limited storage, battery, and processing power to much more powerful ones, it is useful to have a language with subsets corresponding to different levels of hardware capabilities. This can be achieved by limiting the synchronization and call mechanisms.

We have defined virtual machines for different versions of the semantics, defined in rewriting logic. This setting provides prototyping tools as well as model checking tools. Furthermore, it supports distributed heterogeneous networks [4], dynamic software updates, by asynchronous runtime upgrades of code, where each object may upgrade its software independently of other objects [6]. This added layer of semantic complexity caused by dynamic updates may be supported by an operational semantic framework to support modularity of the semantic definition as suggested in [2].

With respect to formal analysis, we have developed methods for static modular analysis of safety, security, and privacy properties, and we have developed tool-oriented verification methods. In particular, compositional reasoning of software updates is supported provided old invariants are

not violated, and reasoning about software evolution is supported even when old invariants are violated [8]. We here want to extend earlier work to include robotic systems and integration of robotic systems and IoT.

References

[1] Martin Carroll, Kedar S. Namjoshi, and Itai Segall. The Resh programming language for multi-robot orchestration. *CoRR*, abs/2103.13921, 2021.

[2] Christian Johansen and Olaf Owe. Dynamic structural operational semantics. *J. Log. Algebraic Methods Program.*, 107:79–107, 2019.

[3] Einar Broch Johnsen and Olaf Owe. An asynchronous communication model for distributed concurrent objects. *Software and Systems Modeling*, 6(1):35–58, March 2007.

[4] Einar Broch Johnsen, Olaf Owe, Joakim Bjørk, and Marcel Kyas. An object-oriented component model for heterogeneous nets. In Frank S. de Boer, Marcello M. Bonsangue, Susanne Graf, and Willem P. de Roever, editors, *Formal Methods for Components and Objects, 6th International Symposium, FMCO 2007, Amsterdam, The Netherlands, October 24-26, 2007, Revised Lectures*, volume 5382 of *Lecture Notes in Computer Science*, pages 257–279. Springer, 2007.

[5] Einar Broch Johnsen, Olaf Owe, Dave Clarke, and Joakim Bjørk. A formal model of service-oriented dynamic object groups. *Sci. Comput. Program.*, 115-116:3–22, 2016.

[6] Einar Broch Johnsen, Olaf Owe, and Isabelle Simplot-Ryl. A dynamic class construct for asynchronous concurrent objects. In Martin Steffen and Gianluigi Zavattaro, editors, *Formal Methods for Open Object-Based Distributed Systems, 7th IFIP WG 6.1 International Conference, FMOODS 2005, Athens, Greece, June 15-17, 2005, Proceedings*, volume 3535 of *Lecture Notes in Computer Science*, pages 15–30. Springer, 2005.

[7] Farzane Karami, Olaf Owe, and Toktam Ramezanifarkhani. An evaluation of interaction paradigms for active objects. *Journal of Logical and Algebraic Methods in Programming*, 103:154 – 183, 2019.

[8] Olaf Owe, Elahe Fazeldehkordi, and Jia-Chun Lin. A framework for flexible program evolution and verification of distributed systems. In *Model-Driven Engineering and Software Development - 7th International Conference, MODELSWARD 2019, Prague, Czech Republic, February 20-22, 2019, Revised Selected Papers*, volume 1161 of *Communications in Computer and Information Science*, pages 320–349. Springer, 2019.

[9] Daniel Simon, Roger Pissard-Gibollet, and Soraya Arias. Orccad, a framework for safe robot control design and implementation. In *1st National Workshop on Control Architectures of Robots: software approaches and issues CAR'06*, Montpellier, France, April 2006.

[10] AdaCore: SPARK. https://www.adacore.com/about-spark.

[11] BehaviorTree.CPP. https://www.behaviortree.dev.

[12] roscon.ros.org. https://roscon.ros.org/2018/presentations/ROSCon2018_rcladatheadaclientlibraryforros2.pdf.

[13] ROS.org. http://wiki.ros.org/smach.

[14] SMAC – state machine asynchronous C++. https://smacc.dev.

[15] Verification and testing of mobile robot navigation algorithms: A case study in spark . https://ieeexplore.ieee.org/abstract/document/6942753?casa_token=3ZtxqJUOtzoAAAAA:d5T2dkigwDdCYuPQZadDcrs0IwCJTwaTdp-UYy-AzyNjJte2CGQepE2tl9c6cbH5qneZ2iPUw0M.