Work-in-Progress: Introducing Assume-Guarantee Contracts for Verifying Robotic Applications

[1]Stefano Spellini, [1,2]Michele Lora, [2]Sudipta Chattopadhyay, [1]Franco Fummi
[1]Dept. of Computer Science, University of Verona – name.surname@univr.it
[2]Singapore University of Technology and Design – name_surname@sutd.edu.sg

***Abstract*—This paper summarizes the first steps toward an automatic framework, relying on Assume-Guarantee Contracts, for the verification of robotics applications. Classic HW and SW design and verification techniques are inadequate for robots due to the involved complexity. In this paper we advocate that contract-based methodologies allow safe problem decomposition easing system-level validation.**

I. Introduction

Robotic systems are becoming every day more safety critical as they exit the research laboratories to enter in the real world [1]. Consequently, reliable design and verification of systems are gaining importance and interest. Traditionally, robots, as well as other families of cyber-physical systems, are validated through simulation [2]: a practice not providing the rigorousness required by safety-critical applications. Recently, formal methods have been introduced in this field [3]. However, formal verification is computationally demanding as it requires to solve exponential problems. This fact is further worsened by the increasing complexity of the application being designed. As such, problem decomposition will be an important feature for any future verification technique that targets robotic systems.

The *Assume-Guarantee (A/G) Contracts* [4] theory allows to decompose system design both horizontally (among different components), and vertically (among different levels of abstraction). An A/G Contract formally represents a component as two sets of behaviors defined over its variables: the *assumptions* and the *guarantees* [4]. A system can be represented as a composition of components at different levels of abstraction and according to diverse points of views. A/G Contracts theory captures these aspects of design perfectly by allowing the definition of the system using multiple contracts, and by defining *horizontal contracts* and *vertical contracts* [5]. Horizontal contracts are supported by the *composition operation* defined by the Contract's algebra, that composes two (or more) contracts into one. Vertical contracts are supported by multiple operations. *Conjunction* composes multiple contracts representing multiple viewpoints of the same component. *Refinement* and *Abstraction* capture the different abstraction-levels of a component. Furthermore, the theory defines *Compatibility* and *Consistency* of a Contracts. A contract is *compatible* if an environment implementing the assumptions can exist. Meanwhile, a contract is said to be *consistent* if an implementation that respects its guarantees is realizable.

In this work, we introduce our contract-based approach to decompose the verification problem into multiple sub-problems that a combination of formal and simulation-based techniques can solve. We intend to build an automatic framework on top of the *CHASE (Contract-based Heterogeneous Analysis and Systems Exploration) library* [6], which provides an API for the representation and manipulation of systems represented by using Assume-Guarantee Contract.

II. Overview of the approach

Figure 1 depicts the structure of our approach. Initially, the robotic system is represented by a set of requirements and the components

This work is partially supported by National Robotics Programme (NRP) – *Robotics Enabling Capabilities and Technologies* (RECT), Grant no. 172 25 00022, and by the *MIUR "Dipartimenti di Eccellenza" 2018-2022* grant.

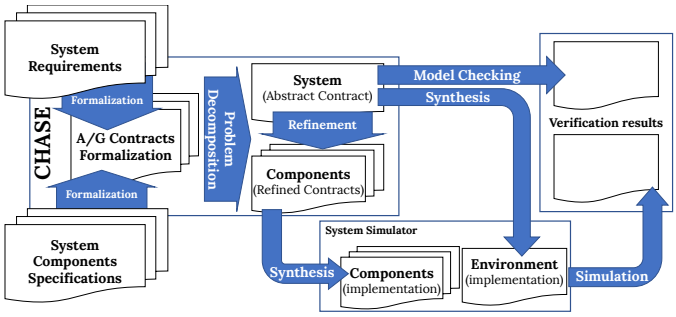

Figure 1: Overview of the proposed approach.

composing the system. The system components and requirements are formalized as a set of A/G contracts. In principle, *the requirements are used to generate the guarantees* of the system. Meanwhile, the *components specifications are mostly used to create the assumptions* of the contract-based description. Intuitively, *the components of the system act as an environment for the requirements*. As such, the system can implement the requirements if it is *consistent*.

Consistency checking of a contract is an exponential problem in general [7]. Thus, we exploit horizontal and vertical contracts to decompose the problem. Abstract instances of the problem are verified by using formal verification, such as model checking. Then, the sub-parts of the abstract specification are refined and synthesized to create their executable models for simulation. Finally, simulation of the entire system is created by aggregating the implementations of the sub-components into a single simulation.

We rely on already well-known techniques for formal verification, such as model checking [8] and reactive synthesis [7]. However, contract-based formalization of components and requirements, as well as contract-based simulation are still unexplored fields.

a) Specification of robotic applications: CHASE provides a flexible mechanism that allows creating front-end tools for different specification languages and input representations, and to formalize as A/G contracts [6]. We are exploiting such features of CHASE to define a *Domain Specific Language (DSL)* for robotic applications. The DSL must be powerful enough to capture the requirements, as well as to describe the features of the system. It must support different kind spatial specifications (occupancy maps, graphs, *etc.* [9]), as well as different timing models (real-time, discrete time, [10]). Furthermore, it must allow specifying robotic requirements, such as task planning, path planning and motion planning objectives. Finally, it must also allow specifying the environment of the system.

b) Simulation-based verification from Contracts: The assumptions of a contract identify the "legal" environments for the system being described. As such, they can be considered as the model of the environment for the system to implement. Furthermore, if the contract representing the system is *compatible* then there exists an implementation of the environment. The simulation technique we are developing exploits this feature. Each sub-component of the system is synthesized by using their refined contracts. Then, the overall contract representing the system is used to generate implementations of the environments automatically. The set of gen-

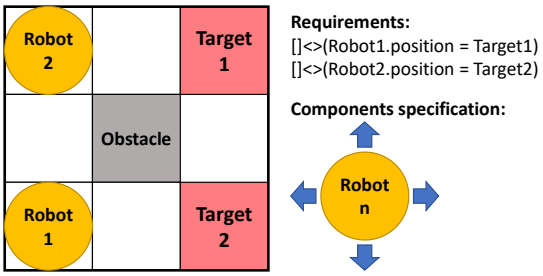

Figure 2: Example of experimental scenario instance.

erated implementations acts as testbenches for the ensemble of sub-components implementations. As such, each agent (*i.e.*, environment and components) involved in the simulation is generated singularly using formal methods. System-level validation, however, is performed by using simulation. Moreover, the approach allows importing models of previously designed components by exploiting state-of-the-art code manipulation techniques for heterogeneous systems [11].

III. Preliminary Results

To evaluate the feasibility of combining assume-guarantee reasoning and simulation, we performed a preliminary experiment. A set of robots moves in a 2-dimensional space, each of the robots must eventually reach a target position. We automatically generated the contract-based specification for scenarios with different dimensions in terms of the number of robots and size of the 2-dimensional space. Each robot can move in four directions and has sensors identifying obstacles in each of the four directions. Figure 2 shows the case of two robots moving in a 3x3 grid.

For each instance of the problem, three contract-based specifications have been created:

- C_S: a single contract representing the entire problem. The representation of the 2-dimensional space is discretized and represented as an occupancy grid [9]. Each robot is represented by four boolean variables representing the four directional sensors, an integer variable for the position, and an integer variable for the robot command. For each robot and each tile of the grid, an assumption assertion specifies the value of the robot sensors.
- C_R: the contract representing a single instance of the robot, assuming the environment. It has four sensors input (one for each direction it can sense), its current position, and its target position. Its assumptions describe the complete information about the environment. For instance, the assertion describing the robot number 2, when its position is the one depicted in Figure 2:

  $$\square(position = 2 \rightarrow (\neg up \wedge down \wedge \neg left \wedge right))$$

  Its guarantees specify how the next command has to be computed. For instance, when the robot number is in the top-left corner:

  $$\square(((position = 2) \wedge \neg up \wedge down \wedge \neg left \wedge right) \rightarrow X(command = go\_down \vee command = go\_right))$$

- C_E: the contract representing the environment (the 2-dimensional space) and assuming the robots. Its inputs are the commands generated by the robots, while its outputs are the robot positions as well as the sensor values of each robot. In this way, the contract represents "the physics" of the system. For each robot, for each position and each command, an assertion is in charge of computing the next position of the robot. For instance, the assertion moving down robot 2:

  $$\square(robot2.position = 2 \vee robot2.command = go\_down \rightarrow X(robot2.position = 1))$$

  Furthermore, C_E also produces as output the target of each robot.

We started by specifying the entire problem in the C_S. Contracts C_R and C_E have been obtained by decomposing C_S. The composition of C_E with a number of instances of C_R equal to the number of the robots in the scenario is equivalent to C_S.

Table I: Model checking versus simulation.

Grid Size	# Robots	# Safety Invariants	Verification Time (s)	
			Reactive Synthesis	Simulation-based
3x3	1	151	0.02	0.03
3x3	2	304	0.27	0.04
5x5	3	1583	2090.16	3.21
5x5	5	2855	Time Out (6 hours)	5.50

The C_R has been refined by removing the assumptions about eventual obstacles. Then, its implementation has been synthesized. as well as the environment contract. We carried out a consistency checking of C_S representing the entire system by using reactive synthesis tools [12]. Finally, we performed the same consistency checking through simulation by using the synthesized implementations of contracts C_R and C_E. Table I shows the results for different sizes of the problem, it reports the number of safety invariants in each specification and compares the time required using formal methods and our simulation environment. Indeed the specifications have been thought to be easily decomposed, and well suited for simulation: thus, justifying the outstanding performance achieved. Still, these results convince us to continue pursuing the road introduced in this preliminary paper.

IV. Conclusions and Future Work

In this paper we introduced our plan to exploit assume-guarantee reasoning and simulation to efficiently verify robotic applications. In the near future we target the definition of efficient and intuitive specification methods for robotics applications. Furthermore, we aim at proposing a complete end-to-end approach decreasing the time required to verify robotic systems. The approach combines formal methods and simulation techniques tied by the A/G contracts theory.

References

[1] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A Survey of Research on Cloud Robotics and Automation," *IEEE Transactions on Automation science and Engineering*, vol. 12, no. 2, pp. 398–409, 2015.

[2] D. Bresolin, L. Di Guglielmo, L. Geretti, R. Muradore, P. Fiorini, and T. Villa, "Open problems in verification and refinement of autonomous robotic systems," in *Proc. of Euromicro DSD*. IEEE, 2012, pp. 469–476.

[3] H. Kress-Gazit, "Robot challenges: Toward development of verification and synthesis techniques," *IEEE Robotics & Automation Magazine*, vol. 18, no. 3, pp. 22–23, 2011.

[4] A. Benveniste *et al.*, "Contracts for system design," *Foundations and Trends® in Electronic Design Automation*, vol. 12, no. 2-3, pp. 124–400, 2018.

[5] P. Nuzzo, A. L. Sangiovanni-Vincentelli, D. Bresolin, L. Geretti, and T. Villa, "A platform-based design methodology with contracts and related tools for the design of cyber-physical systems," *Proceedings of the IEEE*, vol. 103, no. 11, pp. 2104–2132, 2015.

[6] P. Nuzzo, M. Lora, Y. A. Feldman, and A. Sangiovanni-Vincentelli, "CHASE: Contract-based requirement engineering for cyber-physical system design," in *Proc. of IEEE/ACM DATE*, 2018, pp. 839–844.

[7] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive (1) designs," *Journal of Computer and System Sciences*, vol. 78, no. 3, pp. 911–938, 2012.

[8] A. Cimatti, M. Dorigatti, and S. Tonetta, "OCRA: A tool for checking the refinement of temporal contracts," in *Proc. of the 28th IEEE/ACM ASE*. IEEE Press, 2013, pp. 702–705.

[9] I. Gavran, R. Majumdar, and I. Saha, "Antlab: A Multi-Robot Task Server," *ACM Transactions on Embedded Computing Systems*, vol. 16, no. 5s, p. 190, 2017.

[10] A. Desai, T. Dreossi, and S. A. Seshia, "Combining model checking and runtime verification for safe robotics," in *International Conference on Runtime Verification*. Springer, 2017, pp. 172–189.

[11] M. Lora, S. Vinco, and F. Fummi, "A Unifying Flow to Ease Smart Systems Integration," in *Proc. of IEEE HLDVT 2016*. IEEE, pp. 113–120.

[12] I. Filippidis, S. Dathathri, S. C. Livingston, N. Ozay, and R. M. Murray, "Control Design for Hybrid Systems with TuLiP: The Temporal Logic Planning Toolbox," in *Proc. of IEEE CCA 2016*, pp. 1030–1041.