사이버 물리 시스템 오브 시스템즈의 행동 기술 자동화 기법

Esther Cho ○, 김한수, 신용준, 배두환
KAIST 전산학부
{esthercho, hansukim, yjshin, bae}@se.kaist.ac.kr

Automatically Generating Behavior Descriptions of a Cyber-Physical System-of-Systems

Esther Cho ○, Hansu Kim, Yong-Jun Shin, Doo-Hwan Bae
School of Computing, KAIST

Abstract

Cyber-Physical System-of-Systems (CPSoS) is a set of Cyber-Physical Systems (CPSs) that interact with each other and operate in a physical environment. When developing or prototyping such complex systems, engineers make various decisions based on not only the facts in the system and environment but also their assumptions in uncertain cases of the system operation. However, these assumptions may be inaccurate resulting in a gap between expectations and reality, which may result in unintended behavior or failure of the system. To address these issues and to realize realistic knowledge, analyzing uncertainties of CPSoS from field execution is critical in iterative development processes. Therefore, in this study, we propose a search-based method to automatically generate behavior descriptions of CPSoS in the form of a causal relationship of two propositions obtained from field experiment data. In a case study, we applied the proposed method and evaluated the generated behavior descriptions to reveal that they provide useful insights into the target CPSoS for the next cycle of development.

1. Introduction

Cyber-Physical System-of-Systems (CPSoS) is a distributed yet interconnected system of Cyber-Physical Systems (CPSs) interacting with each other while operating in a physical environment [1]. During development, engineers make various decisions based on the assumptions of the system behaviors and the surrounding physical environment [2] that has innate uncertainties [3]. Consequently, engineered CPSoS may not operate as anticipated, resulting in a discrepancy between the assumptions and what transpires in the real world.

This encounter of uncharted situations not only ensues a gap between the assumption and reality, but also can result in unintended behavior or failure of the system. To minimize this gap and obtain realistic knowledge, analyzing uncertainties of CPSoS from field execution is critical in iterative development processes. Consequently, engineers may generate descriptions to capture their knowledge of CPSoS behaviors and its environment for the next iteration in the engineering process. However, engineers' incomplete understanding of the system may bias the descriptions and inaccurately capture the CPSoS behaviors. In addition, engineers cannot generate behavior descriptions of unknown characteristics.

Therefore, in this paper, we applied a search method to automatically find CPSoS behavior descriptions. The descriptions are in the form of a causal relationship of propositions obtained from the field experiment data, which reflects the real behaviors of the CPSoS. Accordingly, found descriptions indicate what actually occurs in the system operation and the surrounding context, which may help engineers to have a better understanding of the system behaviors. Through a case study on a real CPSoS, the proposed search method was evaluated. The generated behavior descriptions revealed useful insights, such as the interactions between the constituent CPSs and the relationships between different actuator variables.

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2022-2020-0-01795) and (No. 2015-0-00250, (SW Star Lab) Software R&D for Model-based Analysis and Verification of Higher-order Large Complex System) supervised by the IITP(Institute of Information & Communications Technology Planning & Evaluation).

2. Background

Genetic Algorithm (GA) is a type of search method that mixes and mutates solution representations (i.e., chromosomes) and applies evolutionary pressure to drive the search for an optimal solution. GA is used to solve complex problems that may be too intricate or impossible to solve using a rule-based solver. Because GA searches from a large search space, proper 1) representation of the chromosomes, 2) designs of genetic operators, and 3) construction of fitness function are crucial to find optimal solutions to a given problem. In this paper, GA is applied to generate behavior descriptions of a CPSoS solely based on the field experiment data.

3. Approach

The overall process for generating CPSoS behavior descriptions is demonstrated in Figure 1, which utilizes GA. After engineers develop a CPSoS, they can execute the system to obtain the field experiment data. The data is a time-series data of the observable variables (e.g., sensing and actuating variables) recorded by each CPSs in the CPSoS. First, the field experiment data is used to generate initial population according to the chromosome representation and to evaluate population by the fitness function. Then, genetic operators are applied to the selected parent population to evolve and generate offspring. The detailed representations of the chromosomes, design of genetic operators, and construction of the fitness function are described in the following sections.

3.1 Chromosome Representation

To generate and evolve CPSoS behavior descriptions, we first represent a behavior description as a causal relationship between two

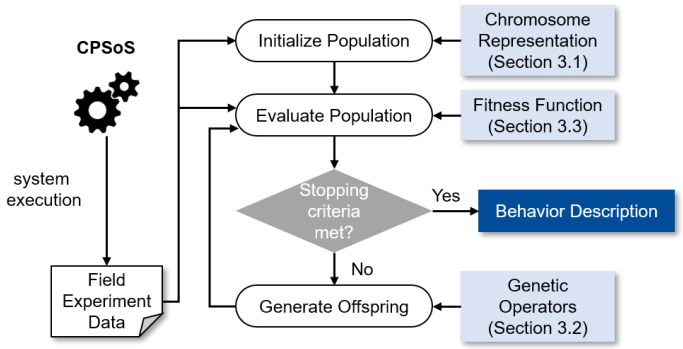

Figure 1: GA Process for Generating Behavior Descriptions

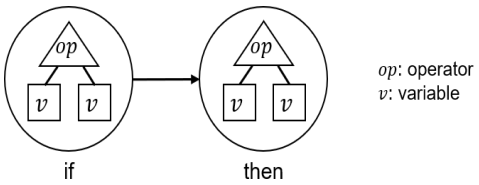

Figure 2: Representation of a Behavior Description

propositions. The cause-and-effect relationship of the propositions demonstrates the flow of time in the CPSoS operation. A proposition contains two variables from the field experiment data and an operator. The operator captures the relationship between the variables and can be represented as $<$, $\leq$, $>$, $\geq$, $==$, and $\neq$. A behavior description structure is visualized in Figure 2. The search space of a description is limited to the number of variables and the time span of each variable from the field experiment data.

3.2 Genetic Operators

To evolve the chromosomes (i.e., behavior description), we design the genetic operators using the chromosome representation defined above. Since a behavior description is composed of propositions, which can further be decomposed into a relationship of variables, the genetic operators are also designed to reflect this hierarchy. Behavior crossover swaps the propositions in two parent descriptions to generate two offspring, as shown in Figure 3. On the other hand, variable-operator crossover exchanges the lower-level elements of the descriptions, such as variables and operators. The variable-operator crossover is illustrated in Figure 4.

The mutation operators are also designed to address the hierarchy of the behavior description. The behavior mutation either randomly generates an entirely new proposition or swaps the two propositions in a behavior description to create a new offspring (Figure 5). Similarly, variable-operator mutation either randomly generates an entirely new variable or operator in a proposition or swaps variables or operators in a behavior description (Figure 6).

3.3 Fitness Function

A fitness function is used to guide the search in GA. The constructed fitness function in Equation 1 evaluates the generated behavior descriptions.

$$score = \frac{tp}{tp + tn} * Pen_{time} * Pen_{cohesion} * Pen_{coupling} \quad (1)$$

tp, or true-positive, counts the number of cases a behavior description is satisfied in the field data. On the other hand, *tn*, or true-negative, counts the number of cases that a behavior description is not satisfied. Therefore, $\frac{tp}{tp+tn}$ quantifies the ratio that the given behavior

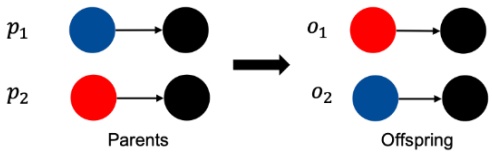

Figure 3: Behavior Crossover

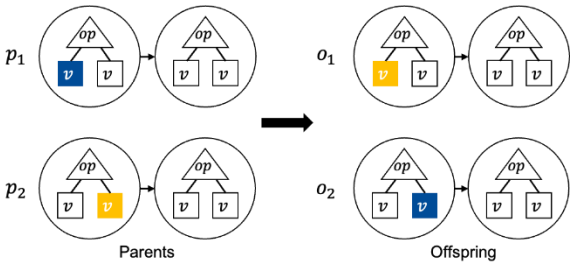

Figure 4: Variable-Operator Crossover

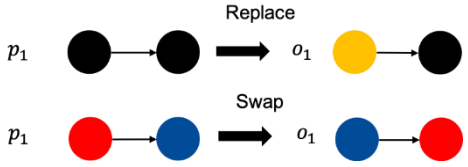

Figure 5: Behavior Mutation

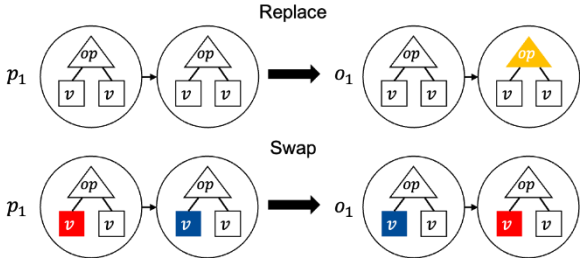

Figure 6: Variable-Operator Mutation

description is really observed in the field experiment.

In addition to the occurrence ratio of the behavior descriptions, the given behavior description was further assessed by introducing time, cohesion, and coupling penalties to guide the search. The three penalty values are between 0 and 1, and thus reduces the overall score even when only one penalty criterion is met.

Time penalty, Pen_{time}, is intended to penalize behavior descriptions with large time gaps. Although a behavior description may be revealed with significant time lag, stating a causal relationship may be impetuous. Consequently, we assign a time penalty to descriptions when the maximum time difference between variables is greater than a certain time threshold. Cohesion penalty, $Pen_{cohesion}$, assesses the relatedness of the variables in a proposition. Propositions containing different variable types are penalized. To retain the diversity of the behavior descriptions, we incrementally increase the cohesion penalty after each iteration of the search. In addition, coupling penalty, $Pen_{coupling}$, evaluates the interdependence among the propositions in a behavior description. Coupling penalty is applied to penalize behavior descriptions with the same system or variable types in two propositions.

4. Evaluation

A case study was performed to evaluate the proposed method on Platooning LEGOs [4], which is an open CPSoS experimental

environment using robot vehicles. The two vehicles each have a role, a leader (Veh 1) and a follower (Veh 2), respectively, and drive on a square track with rounded corners. The implementation of the proposed method can be found in our repository [1]. Experiment settings are shown in Table 1. The generated behavior descriptions are shown in Table 2, which were further scrutinized with respect to the domain knowledge.

Table 1: Experiment Setting

	Parameter	Value
Search	Initial Population Size	200
	Population Size	100
	Crossover Rate	0.6
	Mutation Rate	0.2
	Budget	100
Fitness Function	Max Time Difference	30
	Time Penalty	0.1
	Coupling Penalty	0.9
	Cohesion Penalty Step Size	0.01

Table 2: Generated Behavior Descriptions

	If	Then	Score
1	Veh 1 integral at t+0 ≤ Veh 1 integral at t+5	Veh 2 integral at t+5 ≤ Veh 2 integral at t+6	88.89
2	Veh 1 distance at t+0 ≤ Veh 1 distance at t+4	Veh 1 integral at t+5 ≤ Veh 1 integral at t+6	92.20
3	Veh 2 speed at t+0 ≤ Veh 2 speed at t+20	Veh 2 integral at t+20 ≤ Veh 2 integral at t+21	90.51

The first description is a relationship between integral values of two vehicles. The integral value is a system variable used in the proportional–integral–derivative controller for the adjustment of the steering angle of the vehicle. In our scenario, the integral value increases when the vehicle passes through the corners of the track. The generated behavior description discloses that the follower vehicle enters the corner section at least five ticks after the leader vehicle enters the corner. Thus, this description reveals the relationship between the two CPSs.

The second description is a relationship between distance and integral of the leader vehicle. Since our track was placed parallel to a wall, the distance readings can be reduced when the leader vehicle is approaching the wall. This description shows that after the leader vehicle senses the wall, it begins to turn the corner of the track. Thus, this statement provides insight into our experimental environment that was overlooked.

The third description is a relationship between speed and integral of the follower vehicle. Since the vehicles operated on a squared shape track, the follower vehicle consistently increased its speed at the straight section of the track, while decreasing speed at the corner section. Thus, this statement reflects the characteristic of the system through the relationships of the variable types where the integral value of the vehicle is increased after increase in speed.

In our case study, we applied the proposed approach and found behavior descriptions that provide new and overlooked insights into the CPSoS behaviors and experimental environment. The frequency of the observed behaviors can be described with the score defined in this method. Because the descriptions were generated solely based on the field data, these descriptions appropriately reveal the real behaviors of CPSoS, thus providing knowledge in our system.

[1] https://github.com/est-cho/Belief-Finder

5. Related Work

There are various studies that analyze field experiment data logs of CPS to provide information on the goal-hindering factors, such as anomalies. Krismayer et al. proposed a method for run-time monitoring of the CPS behavior from the CPS event logs to generate system constraints [5]. Liu et al. introduced a path planner that automatically adapts for an autonomous vehicle to avoid collisions based on test logs [6]. Schmidt et al. proposed a method to automatically detect anomalies in CPS behaviors from execution log [7]. These studies focus on analyzing data logs to find incorrect operations of a CPS. On the other hand, our proposed method is centralized on generating behavior descriptions that can describe both correct and incorrect operations.

6. Conclusion

Engineers have preconceived assumptions of a CPSoS behaviors and physical environment in an iterative development process. However, manually capturing these behavior descriptions poses challenges, such as biasing the descriptions and infeasibility to elicit unknown behaviors. To address these challenges, we proposed a search-based method to automatically generate realistic behavior descriptions of a CPSoS from a field experiment data. A case study was conducted on a real CPSoS to show the applicability of the proposed method. The generated descriptions revealed behavior relationships of the system and the environment, which may be used to develop metamorphic relations of a CPSoS for testing.

Reference

[1] S. Engell, R. Paulen, M. A. Reniers, C. Sonntag, and H. Thompson, “Core research and innovation areas in cyber-physical systems of systems,” in International Workshop on Design, Modeling, and Evaluation of Cyber Physical Systems, pp. 40–55, Springer, 2015.

[2] S. Hassan, N. Bencomo, and R. Bahsoon, “Minimizing nasty surprises with better informed decision-making in self-adaptive systems,” in 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp. 134–145, 2015.

[3] Y. -J. Shin, J. -Y. Bae and D. -H. Bae, "Concepts and Models of Environment of Self-Adaptive Systems: A Systematic Literature Review," 2021 28th Asia-Pacific Software Engineering Conference (APSEC), 2021, pp. 296-305.

[4] Y.-J. Shin, L. Liu, S. Hyun, and D.-H. Bae, “Platooning LEGOs: An open physical exemplar for engineering self-adaptive cyber-physical systems-of-systems,” in 2021 International Symposium on Software Engineering for Adaptive and Self- Managing Systems (SEAMS), pp. 231–237, 2021.

[5] T. Krismayer, R. Rabiser, and P. Gru¨nbacher, “A constraint mining approach to support monitoring cyber-physical systems,” in International Conference on Advanced Information Systems Engineering, pp. 659–674, Springer, 2019.

[6] K. Liu, X. Zhang, P. Arcaini, F. Ishikawa, and W. Jiao, “Leveraging test logs for building a self-adaptive path planner,” in Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp. 57–63, 2020.

[7] T. Schmidt, F. Hauer, and A. Pretschner, “Automated anomaly detection in cps log files,” in International Conference on Computer Safety, Reliability, and Security, pp. 179–194, Springer, 2020.