# Belief Finder: Searching for Belief Statements from Empirical Data of Cyber-Physical System-of-Systems

CS454 Team 7
20213597 Meehyun Cho
20213176 Hansu Kim
20170700 Jinu Choi

***Abstract*—Cyber-Physical System-of-Systems (CPSoS) is a set of systems that interact with each other and operate in a physical environment. When engineers develop such complex systems, they make various decisions based on not only the facts in the system environment and contexts but also their assumptions or beliefs in uncertain cases of the system operation. However, these beliefs may be inaccurate resulting in a gap between expectations and reality, which need to be corrected for better decision making in the development of CPSoS. Therefore, in this study, we propose a search-based method to automatically generate belief statements from field experiment data using Genetic Algorithm (GA). In a case study, we applied the proposed method and evaluated the generated belief statements to reveal that they provide useful insights of the target CPSoS.**

## I. Introduction

Cyber-Physical System-of-Systems (CPSoS) is a distributed yet interconnected system of cyber-physical systems (CPSs) communicating 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. This assumption is defined as belief. However, due to the innate uncertainties in CPSoS stemming from the operation in physical environment and interactions among the systems, engineered CPSoS may not operate as anticipated, resulting in a discrepancy between the preconceived belief and what transpires in the real world.

This encounter of uncharted situations not only ensue gap between the belief and reality, but also can result in failure of the system. To prevent influencing design decisions based on inaccurate belief in the system and its environment, analyzing uncertainties of CPSoS in the form of belief is critical. Catering to this need, Belief Improver is being researched, which aims to propose a systematic method for defining and analyzing belief in CPSoS. However, Belief Improver was based on competent engineer hypothesis, where expert knowledge on the developed system is required to generate initial belief statements to utilize the proposed method. Also, the unknown emerging behaviors of CPSoS cannot be defined and analyzed.

Therefore, in this project, we propose a heuristic search method to find belief statements, Belief Finder. By finding belief statements solely based on the experiment data of the engineered CPSoS, expert knowledge and manual efforts in generating initial belief statements are not needed. Found belief statements indicate what actually occurs in the system operation and the surrounding context, and by understanding these beliefs, engineers may have a direction for system improvement. In summary, the main contributions of our approach are as follows:

- We propose a search-based method of generating belief statements of a CPSoS
- We provide a case study of Belief Finder application on a toy example of CPSoS

The remainder of the paper is organized as follows. Section II provides background of the search method used. In Section III, the overall approach and the detailed steps of Belief Finder are introduced. The implementation settings and results of the case study are described in Section IV. Section V discusses the weaknesses of the proposed method and future work. Finally, Section VI concludes the paper.

## II. Background

Genetic algorithm (GA) is a meta-heuristic or optimization search technique motivated by Darwin's theory of evolution, which introduces the principle that population gradually evolves through generational change and that heritable traits survive through breeding and selection. GA can be scrutinized in three parts, representation, recombination, and fitness function. A representation expressing specific solutions are called chromosomes, which are recombined through crossing or mutations to continue generations into new chromosomes. Each chromosome can be evaluated against a fitness value indicating how much it meets the problem conditions, so that more "fit" chromosomes can be transferred to the next generation.

Chromosome representation can encode the appearance, behavior, and qualities of a specific solution. This representation is one of the most important processes in GA because the range of expression and difficulty of recombination depend on it. To design an expressive representation of a solution, various expression methods can be used, such as binary array and tree.

The genetic recombination process mixes and transfers the genes of the parent generation to the child generation effectively, which is done through genetic operators. Crossover operation is important to ensure that new child chromosomes are created by mixing two parents while maintaining some parental characteristics. After crossover, chromosomes undergo a mutation process, which introduces new genes by

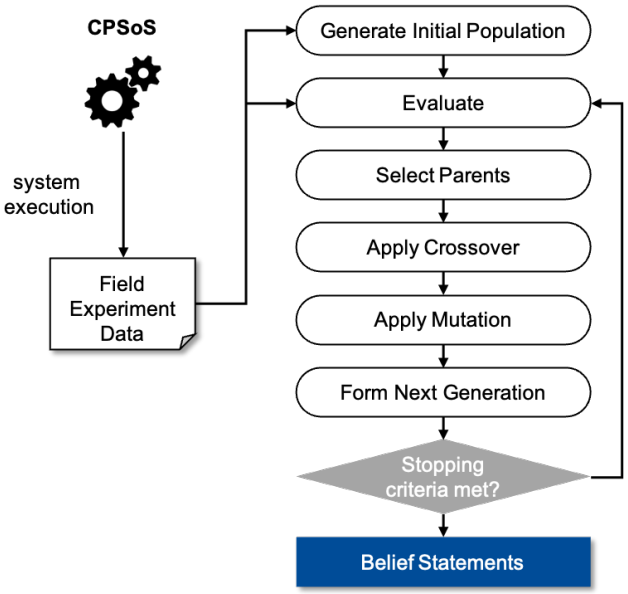



Fig. 1: Overall Approach

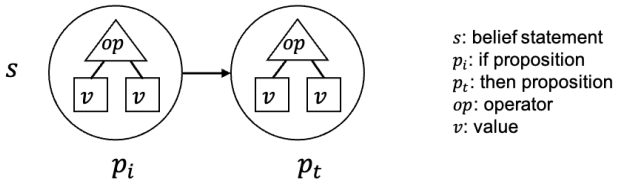
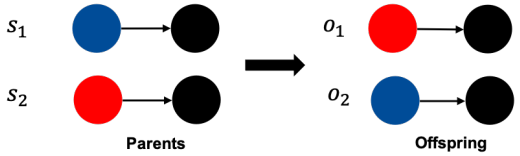
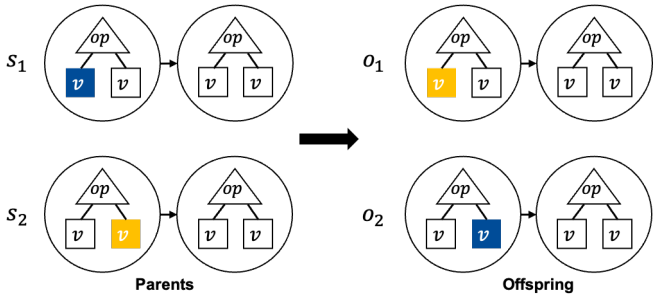



Fig. 2: Representation of a Belief Statement




(a) Proposition Crossover




(b) Value-Operator Crossover

Fig. 3: Crossover Operators

slightly changing original ones for diversity according to a specific probability. In the recombination process, the balance between exploration and exploitation can be adjusted through these two operations. In other words, the process can control the balance between expanding search range of solutions and evolving current solutions.

The fitness function evaluates how an individual is "fit" enough to survive. The survived individuals then pass on their genetic materials to the following generation, while the "unfit" individuals do not get to pass on their genetics. Since GA is a meta-heuristic algorithm, it cannot be designed to suit all problem situations. Therefore, it is important to set the representation, fitness function, and recombination suitable for each problem situation.

## III. Approach

The overall approach is demonstrated in Figure 1, which utilizes GA. After engineers develop a CPSoS, they can execute the system to obtain the field experiment data. Using the proposed method, testers and other engineers without expert knowledge on the engineered CPSoS can automatically find belief statements. First, the CPSoS is executed to obtain field experiment data, which is used to generate initial population and evaluate the population. Then, following the general steps of GA, the population is evolved by iteration of selecting parents and applying genetic operators to generate belief statements. The detailed representations of the chromosomes and genetic operators and a guide for the search are described in the following sections.

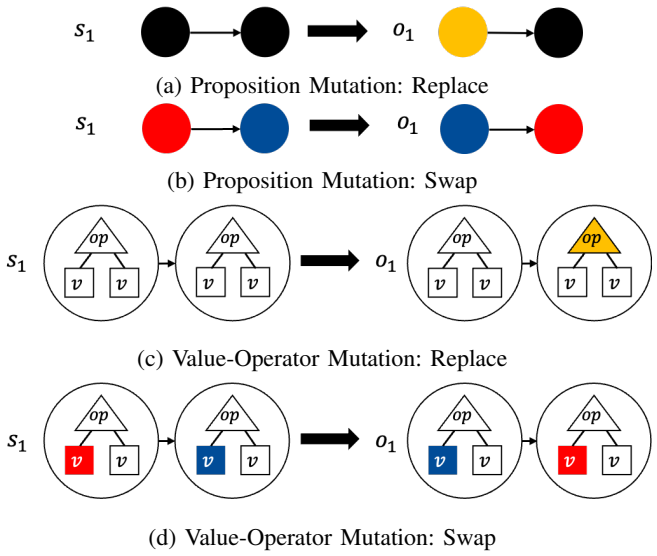
### A. Representation of Chromosomes

To generate and evolve a set of belief statements that capture the behaviors of CPSoS, we borrow the definition of belief statements from our previous work, Belief Improver. Belief statements are defined as a causal relationship between two proposition, such as "if-then" statements. Each proposition is comprised of two values and their relationship, indicated by operators restricted to operators $<$, $\leq$, $>$, $\geq$, $==$, and $\neq$. Each value can either be a variable or a constant, where variables embody time-series data and constants the system invariants, such as threshold values or so-called "magic numbers" in programming. The hierarchical structure of the belief statements can be visualized as shown in Figure 2.

Using the field experiment data, the population of belief statements is generated in which individual statements comply with the hierarchical structure. Also, the field experiment data provide the number of constants and variables, and the time span of each variable. The information on the search space of these values along with the fixed set of operators are utilized to randomly generate individuals forming the population.

### B. Genetic Operators

For the evolution of chromosomes, we design the genetic operators, crossover and mutation, which are defined at different levels of the belief statement hierarchy. Figure 3 depicts the crossover operator. Proposition crossover mixes the proposition of the chromosomes as illustrated by the switch of the red and blue propositions in Figure 3a. Value-operator crossover mixes the genes at a lower-level of the statement, where the values and/or the operators can be mixed as shown in Figure 3b. The mutation operators are depicted




(a) Proposition Mutation: Replace

(b) Proposition Mutation: Swap

(c) Value-Operator Mutation: Replace

(d) Value-Operator Mutation: Swap

Fig. 4: Mutation Operators

in Figure 4. Proposition mutation either introduces an entirely new proposition (Figure 4a) or swaps the two propositions in a statement (Figure 4b). Similarly, value-operator mutation either introduces an entirely new value or operator in a proposition (Figure 4c) or swaps values in a statement (Figure 4d).

### C. Fitness Function

A fitness function is used to guide the search in GA. In Belief Improver, the belief statements were evaluated according to Equation 1.

$$score = \frac{tp}{tp + tn} \tag{1}$$

$tp$ or true-positive is defined as the number of cases where the "if" and the "then" cases are true, while $tn$ or true-negative is defined as the number of cases where the "if" case is true, but the "then" case is false. The $tp$ case implies the causal relationship described by the belief statement is held, while the $tn$ case implies the contrary. However, searching based solely on this equation simplified the generation of belief statement, resulting in more meaningless statements, such as a proposition with two distinct value types with different units and a comparison of values with broad time span.

To address this issue, a modification was made as shown in Equation 2, where time span, cohesion, and coupling factors are considered to guide the search.

$$\begin{aligned} score = & \frac{tp}{tp + tn} \\ & * (1 - Pen_{time}) \\ & * (1 - Pen_{cohesion}) \\ & * (1 - Pen_{coupling}) \end{aligned} \tag{2}$$

Time penalty is intended to penalize belief statements that define relationships of values with large time gaps. Even though a statement may reveal a relationship between characteristics that occur across the dataset, stating a causal relationship may be impetuous. Consequently, we assign a value between 0 and 1 when the max time difference between variables are greater than $n$ ticks and 0 when the max time difference between variables are less than $n$ ticks.

TABLE I: Belief Finder Parameter Settings

| Parameter | Value |
|---|---|
| Initial Population Size | 200 |
| Population Size | 100 |
| Crossover Rate | 0.6 |
| Mutation Rate | 0.2 |
| Budget | 100 |

TABLE II: Fitness Function Penalty Settings

| Parameter | Value |
|---|---|
| Max Time Difference | 30 |
| Time Penalty | 0.9 |
| Coupling Penalty | 0.1 |
| Cohesion Penalty Step Size | 0.01 |

For cohesion and coupling penalties, we borrowed the terms from software engineering, where cohesion measures the relatedness within a module and coupling the interdependence among the modules. To achieve high cohesion of the belief statements, we dynamically penalize propositions containing two different system types, such as system 1 and system 2, or two different data types since the range of values also varies. For example, we cannot compare color reading and speed. To retain diversity of the statements in the beginning, we incrementally increase the cohesion penalty after each iteration. In traditional software engineering, low coupling is desirable to reduce the interdependence among the modules. However, in our approach, coupling penalty is applied to favor statements with different proposition field types when the propositions are cohesive.

## IV. Evaluation

A case study for the Belief Finder was performed to evaluate the proposed method on Platooning LEGOs [2], which is an open physical experimental environment embodying a platooning system, an illustrative example of a CPSoS. In this case study, we used two vehicles, a leader and a follower vehicle, which operated on a squared shape track with rounded corners. The experiment scenario was designed such that the vehicles drove without any obstacle, and thus no lane change occurred to remove anomaly data points.

The field data were collected from data logging by each vehicle, which contained sensor readings, such as color and distance, and actuator values, such as speed. Figure 5 shows the experiment data of vehicle 2. The obtained data is used as an input to our implementation of Belief Finder, which can be found in our repository[1]. Belief Finder was executed using the parameter settings shown in Table I. Generated belief statements during the search are also evaluated against the obtained data using the fitness function, where its penalty settings are organized in Table II.

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

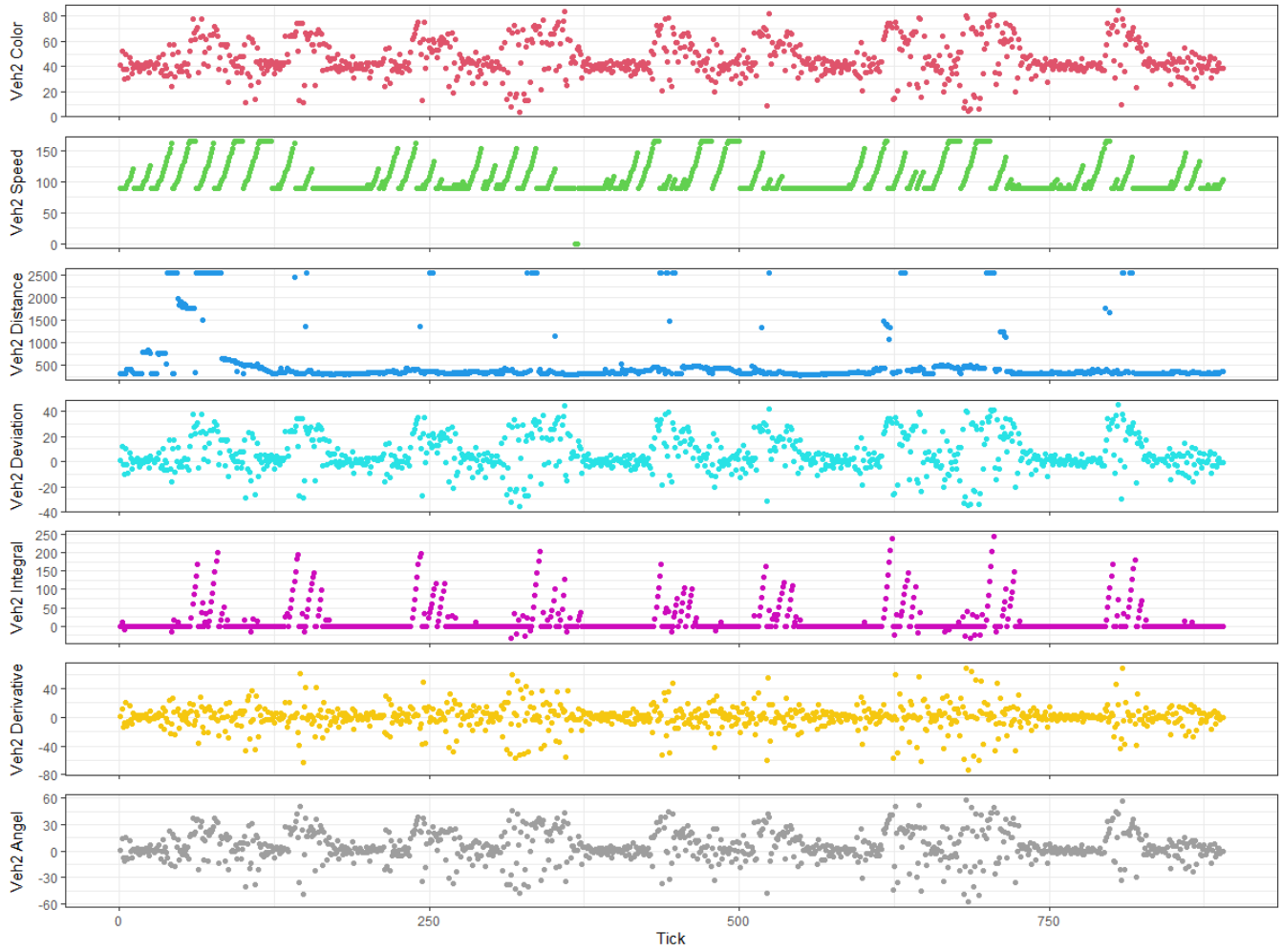



Fig. 5: Case Study Vehicle 2 Field Experiment Data

TABLE III: Generated Belief Statements

| | If | Then | Score |
|---|---|---|---|
| 1 | Veh 1 integral at t+0 $\leq$ Veh 1 integral at t+5 | Veh 2 integral at t+5 $\leq$ Veh 2 integral at t+6 | 88.89 |
| 2 | Veh 1 Distance at t+0 $\leq$ Veh 1 Distance at t+4 | Veh 1 Integral at t+5 $\leq$ Veh 1 Integral at t+6 | 92.20 |
| 3 | Veh 2 Speed at t+0 $\leq$ Veh 2 Speed at t+20 | Veh 2 Integral at t+20 $\leq$ Veh 2 Integral at t+21 | 90.51 |

Once the Belief Finder finished execution, numerous belief statements were generated. Because there were too many generated statements, we manually analyzed them. Also, there were statements that were either meaningless or difficult to analyze, which we further discuss in Section V. From the generated belief statements, we selected three belief statements which may provide useful insights into understanding the system. The selected three belief statements are shown in Table III.

The first statement is a relationship between integral values of two vehicles with a score of 88.89%. The integral value is a system variable used in the proportional–integral–derivative (PID) controller for the adjustment of the steering angle of the vehicle. When the vehicle is turning the corners, the integral value increases. In our scenario, the integral value increases when the vehicle passes through the corners of the track. The generated belief statement discloses that vehicle 2 enters the corner section at least 5 ticks after the vehicle 1 enters the corner. Thus, this statement can reveal the duration for vehicle 2 to reach the current location of vehicle 1.

The second statement is a relationship between distance and integral of the leader vehicle with a score of 92.20%. Since our track is placed parallel to a wall, the distance readings can be reduced when the leader vehicle is nearing the wall. This belief statement shows that after the distance value of leader vehicle is decreased, indicating that the leader vehicle senses the wall, then the vehicle begins to turn the corner of the track. Thus, this statement provides insight of our experimental environment that were not previously considered.

The third statement is a relationship between speed and integral of the follower vehicle with a score of 90.51%. 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 values where the integral value of the vehicle is increased after increase in speed.

In our case study, we applied the proposed approach and found belief statements that provide new insights into our experimental environment. By removing a step to find initial belief statements, we automatically generated statements solely based on the field data, which can be used to understand and improve CPSoS.

## V. Discussion

Although Belief Finder automatically generates statements that reflect the behavior of CPSoS or environmental change, their analysis was still done manually. To address this limitation, we added penalties to those that are likely to be meaningless, such as comparing values of different types, while not removing them from the population to promote diversity in our population.

Additionally, we restricted our data set to only the time-series values (i.e. variables), whereas Belief Improver utilized both the variables and the constants used in the system implementation. When introducing the constants, such as the minimum or maximum values of a data type or thresholds, majority of the generated statements were comparisons of these constants with a score of 100%. Also, we confined the operators depicting the relationships between the values to $\leq, \geq$, as $\leq$ includes relationships covered by $==$ and $<$, and vice versa. We deemed $\neq$ operator useless as most pairs of values indeed have different values.

However, even with these adjustments, generated statements were difficult to analyze or deemed meaningless. Consequently, comparing Belief Finder parameter settings, such as population size and crossover rate, was difficult. Although we executed Belief Finder with various parameter settings, we could not deduce a single best setting for the analysis. Finding a method to analyze and infer meaningfulness of the generated belief statements remains a future work to remove manual analysis and selection of meaningful statements.

## VI. Conclusion

Belief captures preconceived assumptions of the system behaviors and the surrounding physical environment to develop a CPSoS. After the development of CPSoS, field tests are conducted, which provide physical data of the system operation and environment. Belief Improver was proposed to systematically analyze and evaluate belief in CPSoS using the field data. However, Belief Improver required manual efforts and expert knowledge in generating the initial belief statement to run the analysis.

In this project, Belief Finder is proposed to automatically generate belief statements without input belief statement. A case study was conducted on a toy example of a CPSoS to illustrate how Belief Finder can automatically search for belief statements. The generated statements revealed characteristics of the system and the environment. In our proposed method, only GA was used as a search technique. Because system values are likely to exhibit more causal relationships when the time span between the values is closer, applying a local search can assist in an effective finding of belief statements given a combination of value types.

## References


[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*. Springer, 2015, pp. 40–55.

[2] 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)*, 2021, pp. 231–237.