Uncertainty-Wise Model Evolution with Genetic Programming

Man Zhang[1], Shaukat Ali[2,3], and Tao Yue[2]
[1]Kristiania University College, Oslo, Norway
[2]Simula Research Laboratory, Oslo, Norway
[3]Oslo Metropolitan University, Oslo, Norway
man.zhang@kristiania.no, shaukat@simula.no, tao@simula.no

2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security Companion (QRS-C) | 979-8-3503-5939-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/QRS-C60940.2023.00062

Abstract—Model-based Testing (MBT) of a Cyber-Physical System (CPS) under uncertain environments relies on test models manually built based on testers' limited knowledge about the CPS and its operating environment, thereby requiring their continuous evolution. To this end, we propose an uncertainty-wise model evolution approach (UNCERPLORE) to systematically evolve these models with a novel exploration strategy using Genetic Programming while also incorporating CPS execution information. With a preliminary study with a CPS use case, UNCERPLORE manages to evolve models and explore, on average 28.6% new uncertainties in 10 repetitions.

Keywords–model evolution; genetic programming;

1. INTRODUCTION

In a Cyber-physical System (CPS)'s operation, uncertainty exists everywhere, e.g., due to its unpredictable environment. Such uncertainty must be considered during CPS testing. To enable CPS uncertainty-wise testing, we proposed a set of approaches [1], [2], [3], [4], [5] with model and search-based techniques. Among them, UNCERTUM [3] is a modeling framework having *UML Uncertainty Profile* (UUP) and model libraries to create test models (called *Belief Model*s (BMs)) with explicit uncertainty information. A tester creates such models manually based on their understanding of the CPS and its environment. These models require continuous evolution. An example model in UNCERTUM is *Belief State Machine* (BSM) annotated with uncertainty-related model elements modeling uncertain CPS behaviors. An uncertain behavior is observed when triggering the same event from a source state, which could lead to more than one known or unknown state. We propose UNCERPLORE that enables the adaptive exploration of CPS behaviors and supports the evolution of the BM models through direct interactions with the CPS with Genetic Programming (GP).

2. UNCERPLORE

Figure 1 presents UNCERPLORE having three main components, i.e., *Model Execution*, *Model Evolution*, and *Exploration Strategy Evolution*, forming an iterative evolution process. UNCERPLORE takes BSMs as inputs. We define *Exploration Strategy* (ES), which decides how to explore models, e.g., selecting a transition to execute. Guided by ES, *Model Execution* performs the BSM execution with direct communications with the System Under Test (SUT), i.e., the CPS. The model execution enables transitions (e.g., involving an SUT operation) and evaluates the state invariant of the target state based on the SUT status. Based on execution results, *Model Evolution* evolves states, transitions, uncertainties, and uncertainty measurements in the BSM. *Exploration Strategy Evolution* evolves ES for maximizing model coverage and exploring SUT behaviors with GP. The evolved BSM and ES are regarded as inputs of the next iteration of the evolution.

Problem Reformulation. We reformulate ES as a program to guide model execution and allow model evolution by inferring execution information. Figure 1 shows that the ES is modeled as a decision tree with two kinds of nodes: *action* (see *A* in Figure 1) and *condition* (see *C* in Figure 1), and it determines how to explore BSMs under certain conditions. All ES leaves are *action* node. Each action selects the next step to process the model execution, e.g., selecting a transition. Internal nodes define conditions regarding model execution and evolution: 1) the current arrived state in the BSM; 2) the candidates of the outbound transitions; 3) the achieved model coverage; and 4) the achieved uncertainty occurrence coverage.

Model Execution. To enable the BM execution, we defined *executable belief state machines* (ExBSMs). ExBSMs map model elements with variables and accessible SUT test APIs for supporting *state invariant evaluation* and *state transition handling* and collect execution information (e.g., model execution coverage) at run-time. To support adaptive exploration with model execution, the execution of ExBSMs is guided by the specified execution strategy, i.e., ES. The metamodel of ExBSMs is online available[1].

Model Evolution. The model evolves when the current SUT state does not conform to the specified target state after triggering a transition/introducing an indeterminacy source. For instance, as shown in Figure 1, assume that the current state is S_0, after triggering the transition T_1 (selected by ES), the SUT does not reach the state S_1, i.e., the state invariant of S_1 being evaluated as `false`. To identify the new state and uncertainty, we propose Algorithm 1 that derives satisfied constraints based on actual values of potentially related variables of the source target. To infer the state invariant, we define constraint weakening rules specific to Object Constraint Language (OCL) and define collection and numeric operators, i.e., `forAll`, `one`, `includesAll`, `excludesAll`, $=$, $>$, and $<$. In addition, we calculate uncertainty measurement based on the frequency of the occurrence of uncertainty.

Exploration Strategy Evolution with GP. To evolve, UN-

[1]https://github.com/man-zhang/uncerplore

2693-9371/23/$31.00 ©2023 IEEE
DOI 10.1109/QRS-C60940.2023.00062

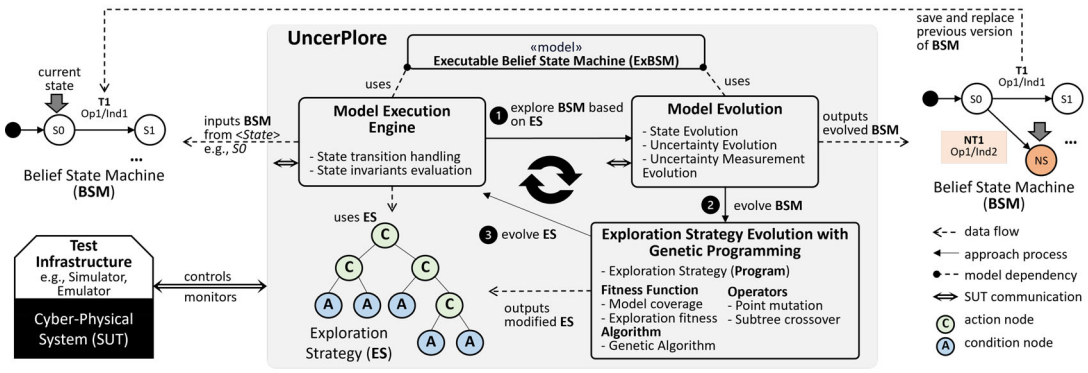

Figure 1. Overview of UncerPlore

Algorithm 1: Identify State and Uncertainty

Input : Specified Target State (tST), Source Target (sST), Triggerred Transition (tTR)
Output: New State (nST), New Uncertainty (nUN)

$nCons \leftarrow \{\}$
$vars \leftarrow getVar(tST)$
for *each variable* $var \in vars$ **do**
  $cons \leftarrow getConstraints(var)$
  for *each constraint* $con \in cons$ **do**
    if $con.evaluate()$ **then**
      $nCons \leftarrow nCons \cup con$
    else
      $wcons \leftarrow weak(con)$
      for *each weaken* $wcon \in wcons$ **do**
        if $wcon.evaluate()$ **then**
          $nCons \leftarrow nCons \cup wcon$
          break
if $|nCons| == 0$ **then**
  $nST \leftarrow createUkST()$
else
  $nST \leftarrow createST(nCons)$
$nUN \leftarrow (sST, tTR, nST)$

CERPLORE employs GP implemented in the MOEA framework [6] with its default configurations, aiming to maximize the execution and model exploration. Thus we define fitness based on the coverage of model execution and the rate of new states and uncertainties detected as $Fitness = 0.2 \times MEC + 0.8 \times Nor(|nSTs| + |nUNs|)$, where *MEC* is the model execution coverage, *nSTs* is a set of newly derived states, *nUNs* is a set of newly discovered uncertainty, $|S|$ represents the size of a set, and *Nor* is a normalization function [7].

3. Preliminary Study

We implemented seven kinds of *condition*s and 11 kinds of *action*s for defining ES to be evolved by GP in UNCERPLORE. We experimented with a preliminary study with one use case of an automated warehouse system. The use case is originally specified with seven uncertainties in its BSM. We set 100 fitness evaluations as GP's stopping criterion and repeated the experiment 10 times. Results show that UNCERPLORE identified on average 2 new uncertainties in the 10-times repetition and evolved 28.6% (2/7) uncertainties.

Acknowledgment

This project is supported by the Co-evolver (286898/F20) and Co-tester (314544) projects from the Research Council of Norway. Man Zhang acknowledges the support from the EAST project (grant agreement No. 864972) from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program.

References

[1] M. Zhang, B. Selic, S. Ali, T. Yue, O. Okariz, and R. Norgren, "Understanding uncertainty in cyber-physical systems: a conceptual model," in *Modelling Foundations and Applications: 12th European Conference, ECMFA 2016*. Springer, 2016, pp. 247–264.

[2] M. Zhang, T. Yue, S. Ali, B. Selic, O. Okariz, R. Norgre, and K. Intxausti, "Specifying uncertainty in use case models," *Journal of Systems and Software*, vol. 144, pp. 573–603, 2018.

[3] M. Zhang, S. Ali, T. Yue, R. Norgren, and O. Okariz, "Uncertainty-wise cyber-physical system test modeling," *Software & Systems Modeling*, vol. 18, no. 2, pp. 1379–1418, 2019.

[4] M. Zhang, S. Ali, T. Yue, and R. Norgre, "Uncertainty-wise evolution of test ready models," *Information and Software Technology*, vol. 87, pp. 140–159, 2017.

[5] M. Zhang, S. Ali, and T. Yue, "Uncertainty-wise test case generation and minimization for cyber-physical systems," *Journal of Systems and Software*, vol. 153, pp. 1–21, 2019.

[6] "MOEA Framework: A Free and Open Source Java Framework for Multiobjective Optimization," http://moeaframework.org/, accessed: August 8th 2023.

[7] V. Vinay, I. J. Cox, N. Milic-Frayling, and K. Wood, "On ranking the effectiveness of searches," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006, pp. 398–404.