EvoMBT at the SBST 2022 Tool Competition

Raihana Ferdous, Chia-kang Hung, Fitsum Kifetew, Davide Prandi, and Angelo Susi
Fondazione Bruno Kessler
Italy
{rferdous,chung,kifetew,prandi,susi}@fbk.eu

ABSTRACT

EvoMBT is a model-based test generator that uses search algorithms to generate tests from a given extended finite state machine (EFSM). In the context of Cyber-physical systems (CPS) testing, and in particular self-driving cars, we model a set of road configurations as an EFSM and use EvoMBT to generate different roads for testing the car. This report briefly introduces EvoMBT and summarizes its results in the Cyber-physical systems testing competition at SBST 2022. Overall the results achieved by EvoMBT are promising where effectiveness and efficiency scores are quite good while the scores related to diversity need improvement.

KEYWORDS

search-based testing, model-based testing, cyber-physical systems, advanced driver assistance systems

ACM Reference Format:
Raihana Ferdous, Chia-kang Hung, Fitsum Kifetew, Davide Prandi, and Angelo Susi. 2022. EvoMBT at the SBST 2022 Tool Competition. In *The 15th Search-Based Software Testing Workshop (SBST'22), May 9, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3526072.3527534

1 INTRODUCTION

EvoMBT [1] is a model-based test generation tool that uses search algorithms to find test suites that achieve different testing goals on a given model. The model is formalized as an extended finite state machine (EFSM) and the test generation is driven by coverage criteria such as state and transition coverage. The abstract tests generated from the model are converted to concrete tests that could be executed against the system under test. While the generation of tests from a given EFSM is quite generic, the concretization of abstract test cases is specific to the system under test. In particular, for the CPS testing tool competition, the abstract test cases are concretized into coordinates on a two dimensional map that correspond to virtual roads for the self-driving car. The objective is to generate valid tests (i.e., virtual roads) that make the car drive out of the boundaries of the road.

This report describes the participation of EvoMBT in the latest edition of the Cyber-physical systems (CPS) testing tool competition at SBST [2] that proposes a case study related to the testing of self-driving cars as the system under test.

EvoMBT performs well on the effectiveness and efficiency categories. This shows that EvoMBT can generate several valid tests under the given time budget. However, EvoMBT did not do well in terms of the diversity score which measures the diversity in the out of bounds revealing tests generated. Future improvements to EvoMBT could be target towards the generation heuristics to maximize the number of fault triggering tests.

In the rest of the paper, we give some technical details about EvoMBT and the model used for the competition. We also comment on the results achieved in more detail.

2 EVOMBT TOOL

EvoMBT is a model based testing tool that uses different algorithms for generating test cases from EFSMs based on a variety of model coverage criteria [1]. An EFSM is a finite state machine (FSM) with internal memory, represented as a set of variables, transition guards and transition effects. When the guard of a transition is true, the corresponding effect is triggered, updating the values stored in the internal variables. More formally, an EFSM E is composed of a set of states S, a set of input values I, a set of output values O, a set of internal variables X, and a set of transitions T. A transition t from state s_i to state s_j has four components: a possibly empty input $i \in I$, a possibly empty output $o \in O$, a boolean guard g with variables in $I \cup X$, and an effect e that updates values of the internal variables. Transition $t = (i, o, g, e)$ is enabled if the guard g is true with the input i and the current value of the internal variables.

EvoMBT is an open source tool and all the code can be found in GitHub: https://github.com/iv4xr-project/iv4xr-mbt.

2.1 BeamNG road model

The CPS testing tool competition requires generating a set of roads where a road consists of a list of points in a 2D space. Each road is evaluated by a driving simulator and failures during simulation are reported. We designed an EFSM such that a path of the model corresponds to a road, i.e., a list of points $(x_0, y_0), \ldots, (x_n, y_n)$.

Fig. 1A shows an example of a test for the CPS. The valid 2D space is defined by coordinates $(0, 0)$, bottom left, and $(35, 25)$, top right. The road starts in $(5, 20)$ and goes through $(15, 20)$, $(15, 10)$, and $(25, 10)$. A road can be alternatively represented as a list of direction vectors from a point to the next one. In Fig. 1A, the road starts at $(5, 20)$, vector v_1 leads to $(15, 20)$, from there vector v_2 goes to $(15, 10)$ and finally v_3 arrives at $(25, 10)$. Fig. 1B sketches how we represent a road with a list of vectors. Given a point (x_i, y_i), the vector leading to next point in the road (x_{i+1}, y_{i+1}) is defined by its direction θ and its magnitude m. Direction θ represents the degree from the x axis, while magnitude m is the length of the vector. The road in Fig. 1A results in vectors $(0, 10)$, $(270, 10)$, and $(0, 10)$.

SBST'22 , May 9, 2022, Pittsburgh, PA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9318-8/22/05…$15.00
https://doi.org/10.1145/3526072.3527534

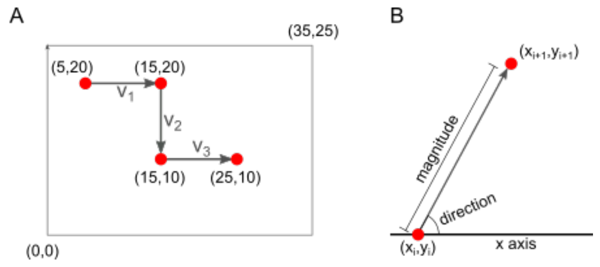

Figure 1: Example of CPS road

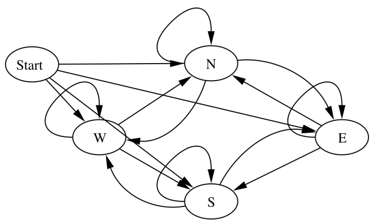

Figure 2: Example of CPS EFSM model

Our EFSM E_{CPS} for CPS relies on vector representation of a road to model the possible directions the road could take. To start, we hypothesize four possible directions, 0, 90, 180, and 270 and a single magnitude, 10, leading to four direction vectors: east $E = (0, 10)$, north $N = (90, 10)$, west $W = (180, 10)$, and south $S = (270, 10)$. Fig. 2 depicts states and transitions of E_{CPS}. States encode possible direction vectors (N, S, E, and W, in this example) plus the initial state $Start$. A transition from a state s_i (direction) to another state s_j reads as: last direction was s_i, now take direction s_j. Clearly, given the last step taken not all directions are possible. For instance, in Fig. 2 it is not possible to go from E to W, as this corresponds to a 180° turn that leads to an invalid road. EFSM E_{CPS} has internal variables (x, y) to track the current position in the 2D space. Guards check that executing the transition does not generate an out of bound, i.e., the values (x, y) are within the limit of the 2D space. Finally, transition effects update (x, y) accordingly with direction and magnitude of the target state. With model of Fig. 2, the road depicted in Fig. 1A traverses states $Start$, E, S, and E.

The model in Fig. 2 is generalized to represent more directions and more magnitudes, making the model representative of a variety of roads. After empirical evaluation, we selected four possible magnitudes, 10, 20, 30 and 40, direction every 30°, and initial position $(40, 40)$. Another parameter control the maximum direction. In Fig. 2, the maximum direction is 90° as, given a state s, next states are s (direction 0°), 90°, and −90°. For instance, next states of E are E (0°), N (90°), or S (−90°). Model E_{CPS} has maximum allowed direction of 45°. With these parameters, E_{CPS} has 13 states and 192 transitions. With a budget of 60s, EvoMBT covers 23.8% of the transitions, averaged over 10 runs. Finally, the generated roads are filtered if too short, too sharp, self-intersecting, etc.

3 BENCHMARK RESULTS

The testing infrastructure uses a single flat road with the fixed weather condition (sunny, without fog). The goal of the test generator is to generate road points that make the car go *out of bounds* (OOBs) and do not violate the following conditions [2]: (i) do not self-intersect; (ii) do not contain turns that are too sharp to be driven without leaving the lane; and, (iii) are fully contained in the map. Two experimental setups are used: BeamNG.AI and Dave-2. BeamNG is the default built-in driving agent. Dave-2 uses deep learning, consisting of 3 convolutional layers and 5 fully-connected layers to predict the driving angles, using camera images that exclude out-of-lane images as training data. The evaluation of the tools is performed along three criteria: test generation *efficiency*, *effectiveness*, and failure-inducing test *diversity* [2]. Table 1 summarizes test generation results for EvoMBT, averaged over 10 runs. It shows number of valid/invalid tests, how many of those passed/failed, and how much time was spent by EvoMBT on generation and execution.

Table 1: Test generation results for EvoMBT

Env	Valid/Invalid	Passed/Failed	Gen/Exec Time (sec)
BeamNG.AI	27 / 0	26.2 / 0.8	3151.42 / 1064.02
Dave-2	23.13 / 0	23.13 / 0	3066.25 / 2351.10

Table 2: Scores achieved by EvoMBT for each criteria

Env	Efficiency	Effectiveness	Diversity	Final Score
BeamNG.AI	<0.001	0.2	0.016	0.216
Dave-2	<0.001	0.2	0	0.2

Test Generation Efficiency and Effectiveness. Efficiency is measured as the time spent to generate tests, normalized over all the runs of all the tools in the same configuration. As reported in Table 2, EvoMBT scored <0.001 for efficiency. Effectiveness is measured as the ratio of valid tests over all the generated tests. EvoMBT's effectiveness score is among the highest at 0.2 (see Table 2). EvoMBT uses the fine-tuned step size, rotation limits and fixed angle transitions to avoid generating overly-sharp turns, and self-intersecting roads. The coordinates of the road points are constrained at a certain range to avoid going outside of the map. Other invalid tests are filtered out during test generation.

Failure-inducing Test Diversity Diversity score calculates the number of different out-of-bound-failures triggered by a road segment. There are two features that characterize the failure: direction coverage and standard deviation of the steering angle [2]. As shown in Table 2, EvoMBT's score with respect to the diversity metric was low. A possible reason for the low diversity score could be that the execution budget was not appropriately used by EvoMBT, probably due to an implementation issue. This is to be investigated. Part of our future work will be to improve EvoMBT's diversity score.

ACKNOWLEDGMENTS

The work presented is part of the iv4XR H2020 project funded under grant number: 856716

REFERENCES

[1] Raihana Ferdous, Fitsum Kifetew, Davide Prandi, ISWB Prasetya, Samira Shirzadehhajimahmood, and Angelo Susi. 2021. Search-Based Automated Play Testing of Computer Games: A Model-Based Approach. In *Search-Based Software Engineering - 13th International Symposium, SSBSE 2021, Bari, Italy, October 11-12, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 12914)*. Springer, 56–71.

[2] Alessio Gambi, Gunel Jahangirova, Vincenzo Riccio, and Fiorella Zampetti. 2022. SBST Tool Competition 2022. In *15th IEEE/ACM International Workshop on Search-Based Software Testing, SBST 2022, Pittsburgh, PA, USA, May 9, 2022.*