Search-Based Construction of Finite-State Machines with Real-Valued Actions: New Representation Model

Igor Buzhinsky
St. Petersburg National Research University of Information Technologies, Mechanics and Optics
Kronverksky pr., 49
St. Petersburg, Russia
buzhinsky@rain.ifmo.ru

Vladimir Ulyantsev, Fedor Tsarev
St. Petersburg National Research University of Information Technologies, Mechanics and Optics
Kronverksky pr., 49
St. Petersburg, Russia
{ulyantsev, tsarev}@rain.ifmo.ru

Anatoly Shalyto
St. Petersburg National Research University of Information Technologies, Mechanics and Optics
Kronverksky pr., 49
St. Petersburg, Russia
shalyto@mail.ifmo.ru

ABSTRACT

In this paper a search-based method for constructing finite-state machines (FSMs) with continuous (real-valued) output actions is improved. A more flexible FSM representation model is presented and compared with the previous one on the problem of unmanned aircraft control.

Categories and Subject Descriptors

I.2.2 [**Artificial Intelligence**]: Automatic Programming—*Program synthesis*

Keywords

Finite-State Machine; Finite-State Machine Construction; Genetic Algorithms; Ant Colony Optimization

1. INTRODUCTION

Finite-state machines (FSMs) are widely used in the development of reactive systems [3, 4, 6]. In this paper we deal with the problem of FSM construction using tests [7]. Specifically, we use the *unmanned aircraft model* as an example of a controlled object. A recently proposed method [1] which exploited genetic algorithms and allowed to learn FSMs with continuous output actions is improved by the introduction of a new, more flexible FSM representation model.

2. PROBLEM STATEMENT

An FSM is a sextuple $(S, s_0, E, A, \delta, \lambda)$ where S is a finite set of states, $s_0 \in S$ is a start state, E is a set of input events, A is a set of output actions, $\delta : S \times E \to S$ is a transition function, and $\lambda : S \times E \to A$ is an output function. On each time step the FSM receives an input event, generates an output action according to λ and changes its active state according to δ.

Consider a set of N tests which describe the proper behavior of a controlled object. A test consists of the input data describing the object's state in different moments and the output data which shows how the object should be controlled. Formally, a test is formed by two sequences: $\text{in}[i] = (\text{in}[i][t])_{t=1}^{\text{len}[i]}$, a sequence of *input tuples*, and $\text{out}[i] = (\text{out}[i][t])_{t=1}^{\text{len}[i]}$, a sequence of *output tuples*, where $\text{len}[i]$ is the test length. An example of a test is shown in Table 1. We consider the problem of learning FSMs which control the object in an appropriate way.

Table 1: An example of a test ($\text{len}[i] = 235$).

Values	Description	$t = 1$	...	$t = 235$
in$[i][t][1]$	Pitch angle (°)	3.078	...	2.412
in$[i][t][2]$	Roll angle (°)	−0.076	...	1.759
in$[i][t][3]$	Yaw angle (°)	198.03	...	205.64
in$[i][t][4]$	Airspeed (knots)	251.42	...	289.40
out$[i][t][1]$	Aileron position	0.000	...	−0.003
out$[i][t][2]$	Rudder position	0.000	...	−0.001
out$[i][t][3]$	Elevator position	−0.035	...	−0.011

Tests can be obtained from a human expert. We used the *FlightGear* flight simulator to record the aircraft's flight data while it was performing aerobatic figures. The input parts of our tests consist of flight parameters: altitude, pitch angle, etc., whereas the output parts are formed by aircraft control (elevator, ailerons, etc.) positions characterized by real numbers (the values of j-th control positions are bounded with $c_j^{\min}$ and $c_j^{\max}$).

3. FSM REPRESENTATION

In the FSM representation model used in [1] there were $2m$ transitions from each state, where m is the number of predicates – Boolean functions of flight parameters. Each time step consisted of m transitions executed according to the values of different predicates. The resulting output tuple on the time step was the element-wise sum of all output tuples of transitions executed during it.

One of the problems of this representation is the complexity of the transition function: the number of values stored for its representation becomes huge with the increase of the number of predicates. Besides, there is no way to take into account the real-valued data provided in the input tuples fully as predicates have Boolean values.

In the approach proposed in this paper, only one transi-

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.
ACM 978-1-4503-1964-5/13/07.

tion is executed during the FSM time step. The transition function δ and output function λ are stored independently. For transition function representation we use the reduced table method proposed in [5]: a predicate *significance mask*, which defines the predicates on which the transition function values depend, is stored in each state. For each combination of significant (marked with 1 in the mask) predicate values, a value of the transition function is stored in the same state. The output actions are formed for different controls as linear combinations of *variable* values. Variables, like predicates, are functions of controlled object parameters, but are real-valued. A mask of significant variables is stored for each state and control.

To reduce the search space during search optimization, we use FSM *skeletons* (FSMs with only transition function and significance masks defined) as individuals.

4. FSM CONSTRUCTION METHOD

A genetic algorithm (as in [1]) and the ACO-based algorithm [2] are used for searching FSMs with the fitness value high enough to perform an aerobatic figure described in a test set. Let $\mathrm{ans}[i] = (\mathrm{ans}[i][t])_{t=1}^{\mathrm{len}[i]}$ be the output sequence produced by an FSM in response to $\mathrm{in}[i]$. A slightly modified version of the fitness function from [1] is used:

$$f = \left(1 - \sqrt{\frac{1}{N}\sum_{i=1}^{N}\frac{1}{\mathrm{len}[i]}\sum_{t=1}^{\mathrm{len}[i]}\frac{1}{C}\sum_{j=1}^{C}\left(\frac{\Delta[i][t][j]}{c_j^{\max}-c_j^{\min}}\right)^2}\right) - P_\tau,$$

where $\Delta[i][t][j] = \mathrm{ans}[i][t][j] - \mathrm{out}[i][t][j]$, C is the output tuple dimension and P_τ is an additional penalty for the number of times the FSM changes its state during execution on tests.

Having an FSM skeleton, it is possible to maximize [1] the fitness function by assigning the values required to define the output function. The output derivation procedure can be generalized to handle the new FSM representation model. It is executed before each fitness function evaluation.

5. EXPERIMENTAL EVALUATION

A comparison between the former and the new models of FSM representation was performed on two test sets which describe the barrel roll aerobatic figure and a 180° turn in the horizontal plane. Both test sets were recorded using the model of the Gloster Meteor jet fighter.

Using the previously developed FSM representation model we have found a predicate set sufficient for the appropriate aircraft control during the barrel roll, but we did not manage to find a proper predicate set for the turn, whereas finding proper predicate and variable sets for the new representation model was not a problem. Table 2 shows the results of running optimization algorithms 50 times on three problem instances: barrel roll with the former model (1), barrel roll with the new model (2), turn with the new model (3). We used *Intel Core 2 Quad Q9400* processor for computational experiments and searched for FSMs with four states.

The quality of some FSMs generated by the search optimization algorithms was examined in computer simulation. After simulating a number of FSMs with different fitness values, FSMs performing both aerobatic figures were found. However, the quality of FSMs with the new representation model was better: the best FSMs with the former representation model were not perfect at the end of the barrel roll. A video record of a flight of the Gloster Meteor performing the barrel roll under control of one of the FSMs with the new representation model is available at http://youtu.be/g5P3UsB0CWI.

Table 2: Results of running the genetic algorithm (GA) and the ACO-based algorithm (ACO) on different problem instances.

Problem instance	(1)	(2)	(3)
Number of fitness evaluations in a run	40000	20000	20000
Average fitness at the end of run, GA / ACO	0.9850 / 0.9858	0.9859 / 0.9862	0.9899 / 0.9899
Average run time, GA / ACO, minutes	11 / 14	10 / 10	16 / 15

6. CONCLUSION

The new model of FSM representation has been developed. It allows to use real-valued variables to form the output actions of FSMs. The new model was compared with the former one on the unmanned aircraft control problem. The former model did not work on one of the two test sets. The use of the new model for the second test set improved the quality of generated FSMs.

7. REFERENCES

[1] A. Alexandrov, A. Sergushichev, S. Kazakov, and F. Tsarev. Genetic algorithm for induction of finite automata with continuous and discrete output actions. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation (GECCO '11)*, pages 775–778. ACM, 2011.

[2] D. Chivilikhin and V. Ulyantsev. Learning finite-state machines with ant colony optimization. In *Lecture Notes in Computer Science*, volume 7461/2012, pages 268–275. Springer, September 2012.

[3] D. Harel and A. Pnueli. On the development of reactive systems. *Logic and Models of Concurrent Systems*, pages 477–498, 1985.

[4] D. Harel and M. Politi. *Modeling Reactive Systems with Statechart. The Statemate Approach.* McGraw-Hill, NY, 1998.

[5] N. Polikarpova, V. Tochilin, and A. Shalyto. Method of reduced tables for generation of automata with a large number of input variables based on genetic programming. *Journal of Computer and Systems Sciences International*, 49(2):265–282, 2010.

[6] A. Shalyto. Logic control and reactive systems: Algorithmization and programming. *Automation and Remote Control*, 62(1):1–29, 2001.

[7] F. Tsarev. Method of finite state machine induction from tests with genetic programming. *Information and Control Systems (Informatsionno-upravljajuschie sistemy, in Russian)*, (5):31–36, 2010.