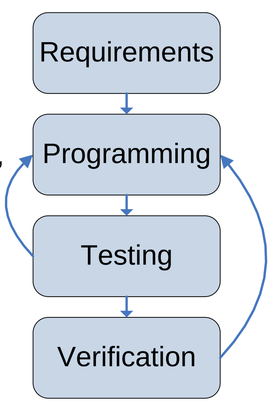
Inferring Automata-Based Programs from Specification With Mutation-Based Ant Colony Optimization

Daniil Chivilikhin and Vladimir Ulyantsev
ITMO University
Computer Technologies Laboratory
Saint Petersburg, Russia

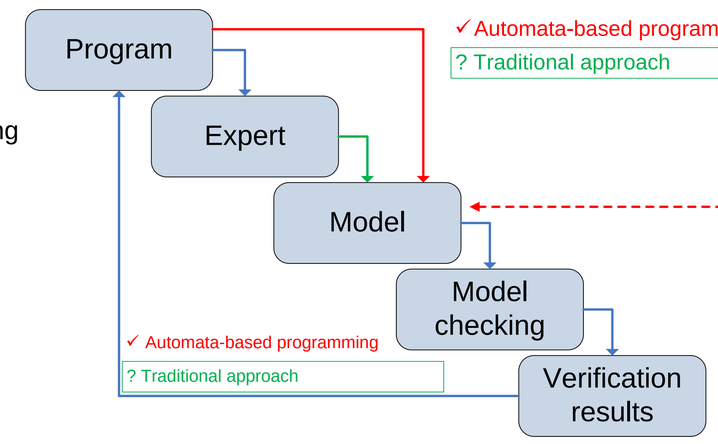
Reliable control system development

Traditional approach

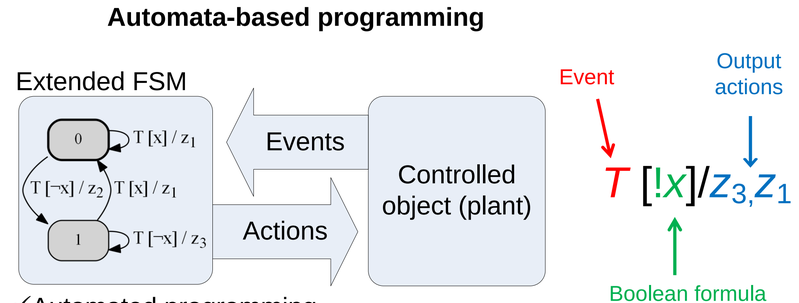


- Certain applications require high software reliability
- Ex: aviation, space industry, banking, energetics
- Verification is needed
- Complex development process

- Manual programming
- Manual model construction
- Model ≠ Program
- Hard to interpret counterexamples

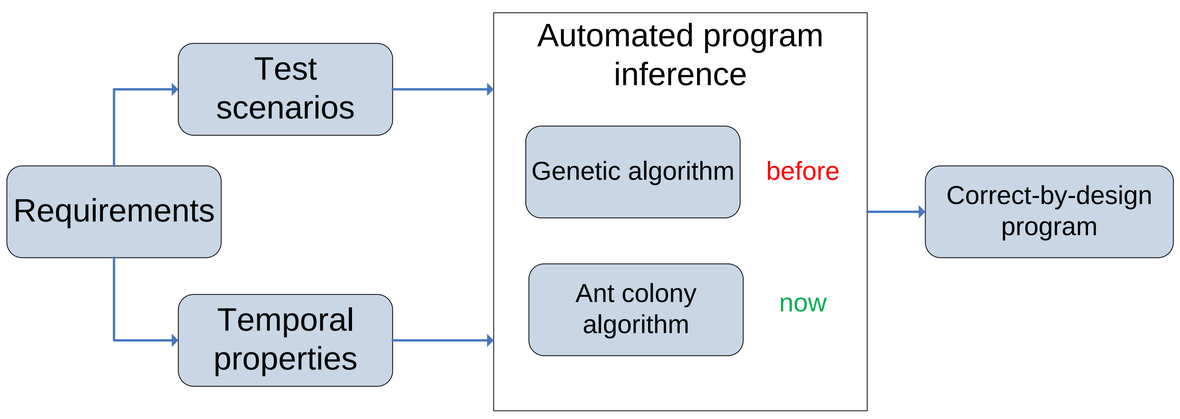


Automata-based programming



- ✓Automated programming
- ✓Automated model construction
- ✓Model = Program
- ✓Automated counterexample interpretation

Inferring automata-based programs



Example: elevator doors control

Events

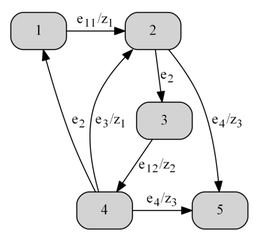
- e_{11} – user pressed “Open doors”
- e_{12} – user pressed “Close doors”
- e_2 – doors opened
- e_3 – obstacle prevents doors from closing
- e_4 – doors are jammed

Actions

- z_1 – start opening doors
- z_2 – start closing doors
- z_3 – call emergency

Scenario	Meaning
e_{11}/z_1, e_2, e_{12}/z_2, e_4/z_3	Doors opened and closed with no problems
e_{11}/z_1, e_{12}/z_2, e_2, z_1, e_2, e_{12}/z_2, e_2	Obstacle prevented doors from closing, closed at second try

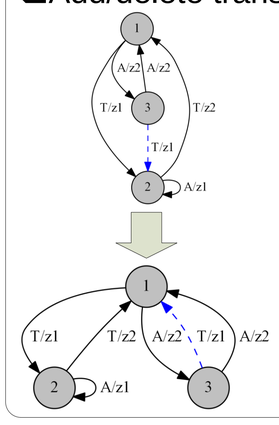
LTL formula	Meaning
G(wasEvent(e_3) ⇒ wasAction(z_1))	If obstacle prevents doors from closing, they will start opening



Mutation-Based Ant Colony Optimization

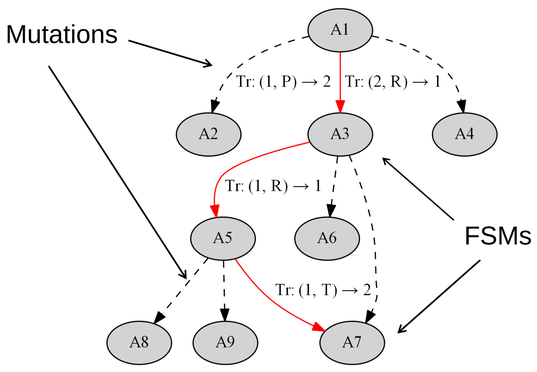
FSM mutations:

- Change transition end state
- Add/delete transition



Mutation graph:

- Nodes = Finite-State Machines
- Edges = FSM mutations



Algorithm

while (true)

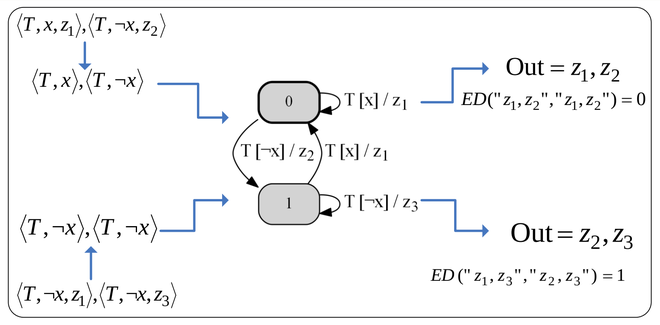
- Build solutions with ant colony
- Pheromone update
- Check stop criteria

Pheromone update

- Update τ_{uv}^{best}
- $\tau_{uv} = \max\left(\tau_{\min}, \rho\tau_{uv} + \tau_{uv}^{best}\right)$

Fitness function

- $T=\{T_i\}_i$ – test scenarios
- Scenario $T_k = \left\langle e_0^k, f_0^k, O_0^k \right\rangle, \ldots, \left\langle e_{n_k}^k, f_{n_k}^k, O_{n_k}^k \right\rangle$
- e – input event
- f – Boolean formula over input variables
- O – sequence of output actions
- Temporal properties (LTL)



string edit distance

# of verified transitions

$$F = \frac{1}{|T|}\sum_{i=1}^{|T|}\left(1 - \frac{\mathrm{ED}(\mathrm{Out}_i, \mathrm{Ans}_i)}{\max\left(\mathrm{len}(\mathrm{Out}_i), \mathrm{len}(\mathrm{Ans}_i)\right)}\right) + \frac{1}{n_{\mathrm{LTL}}}\sum_{i=1}^{n_{\mathrm{LTL}}}\frac{t_{\mathrm{checked}}^i}{t_{\mathrm{reachable}}}$$

total # of transitions

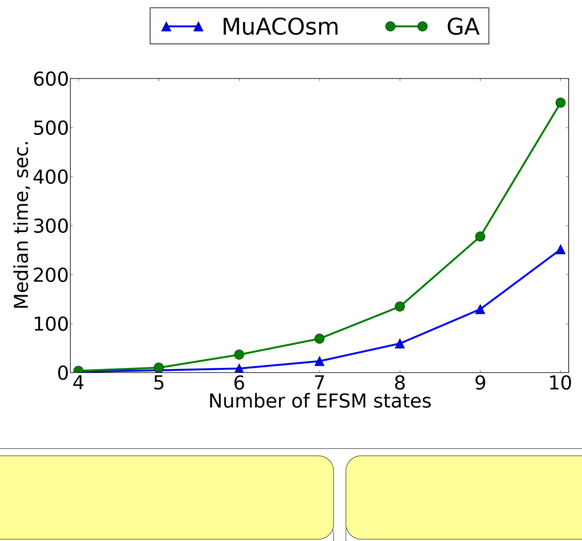
Empirical study

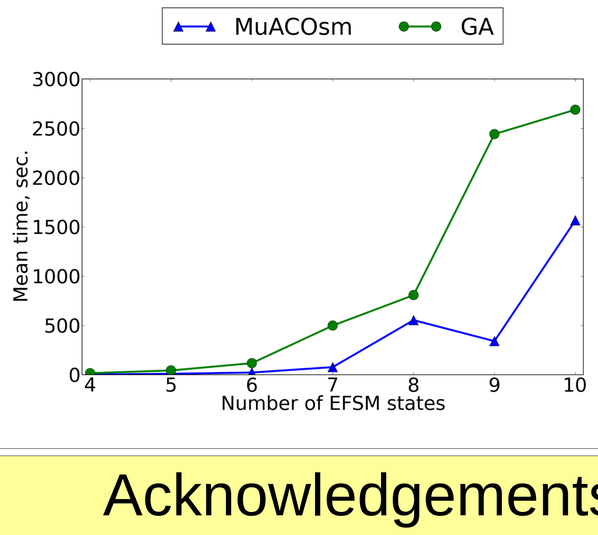
Experiment

- Compare MuACO with previously used GA
- Automated parameter tuning using the *irace* package
- 12 hours for tuning each algorithm

Setup

- N_{states} = 4..10
- Total scenarios length = $100 \times N_{\mathrm{states}}$
- 50 instances for each value of N_{states}
- 2 LTL formulae for each instance





- Wilcoxon statistical test
- Null hypothesis: distributions have the same means

# of states	p-value
4	1e-4
5	6e-4
6	1.35e-8
7	0.24
8	2e-3
9	2e-5
10	3e3

Publications

- Chivilikhin D., Ulyantsev V. MuACOsm - A New Mutation-Based Ant Colony Optimization Algorithm for Learning Finite-State Machines / In Proceedings of the fifteenth Genetic and Evolutionary Computation Conference (GECCO'13), Christian Blum (Ed.). ACM, New York, NY, USA, 2013, pp. 511-518
- Ulyantsev V., Tsarev F. Extended Finite-State Machine Induction using SAT-Solver / Proceedings of the 14th IFAC Symposium “Information Control Problems in Manufacturing - INCOM'12". IFAC, 2012, pp. 512–517

Acknowledgements

This work was financially supported by:

- the Government of Russian Federation, Grant 074-U01;
- RFBR, research project No. 14-07-31337 mol_a.

{chivdan,ulyantsev}@rain.ifmo.ru
http://rain.ifmo.ru/~{chivdan,ulyantsev}
Lab site: http://irc.ifmo.ru/en/87845/