ITMO UNIVERSITY

GECCO

Inferring Temporal Properties of Finite-State Machines with Genetic Programming

Daniil Chivilikhin
PhD student
ITMO University

Ilya Ivanov
Undergrad student
ITMO University

Anatoly Shalyto
Dr. Sci., professor
ITMO University

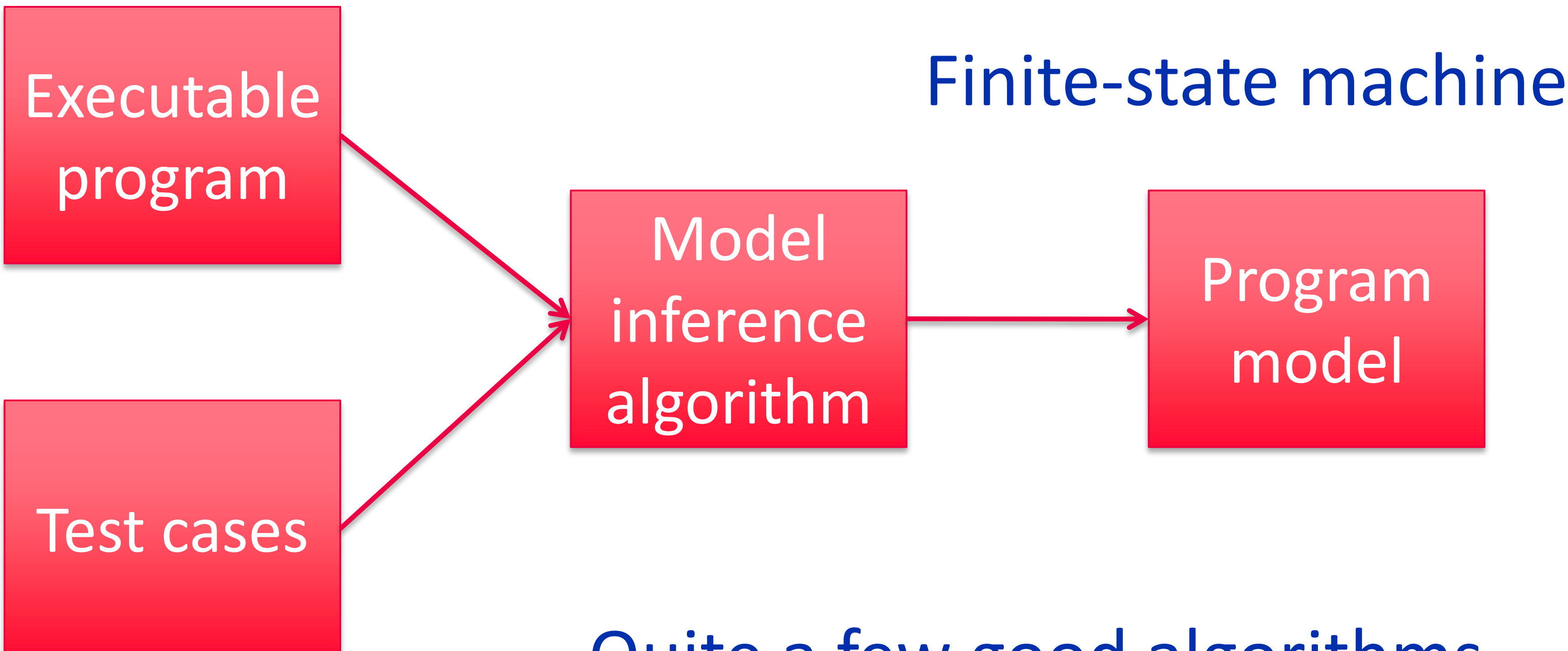
GECCO'15 Student Workshop

July 11, 2015

Introduction

- Software models
- Not always created
- If created, not always kept up to date

Model inference



Quite a few good algorithms

Temporal logics

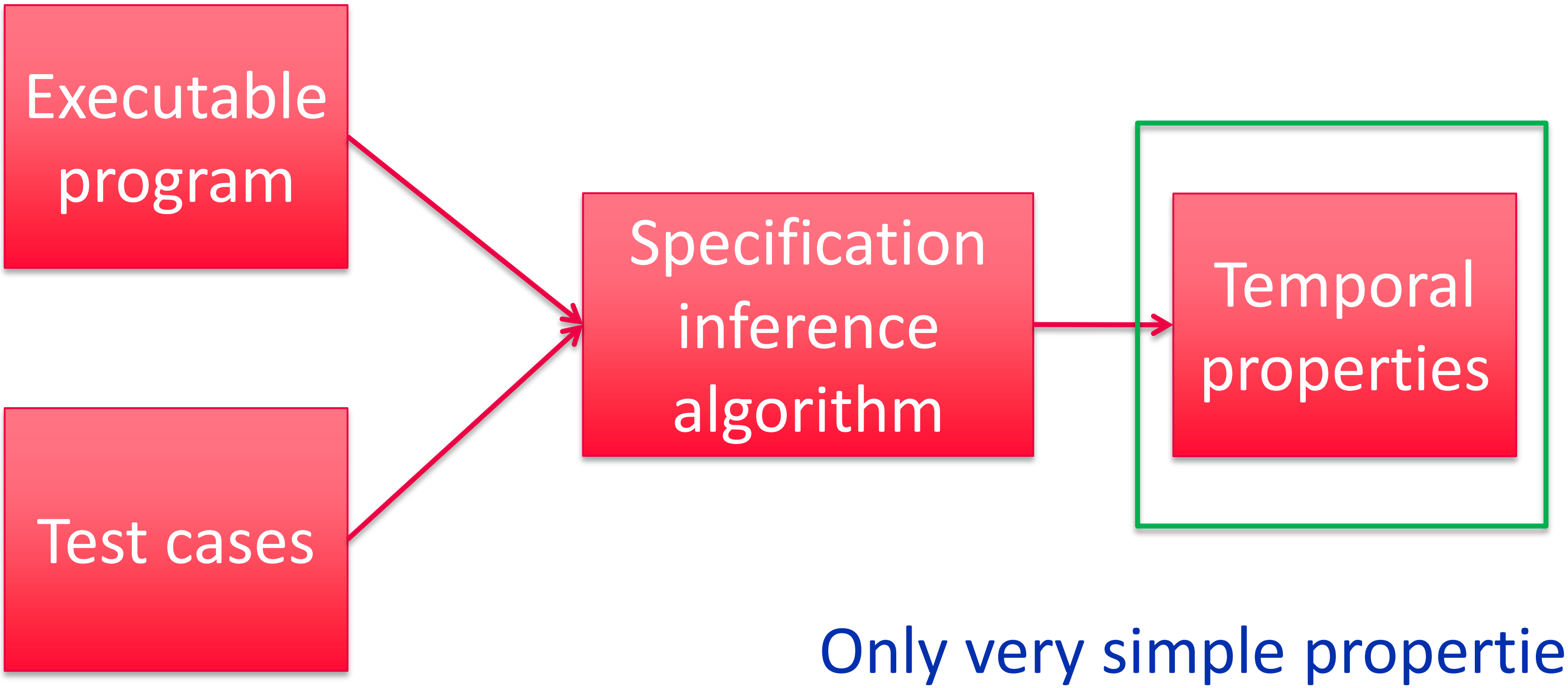
- Used to express time-related propositions
- In software verification: state requirements for software systems
- Example statement

“If a request is received, an answer is eventually generated”

Linear temporal logics

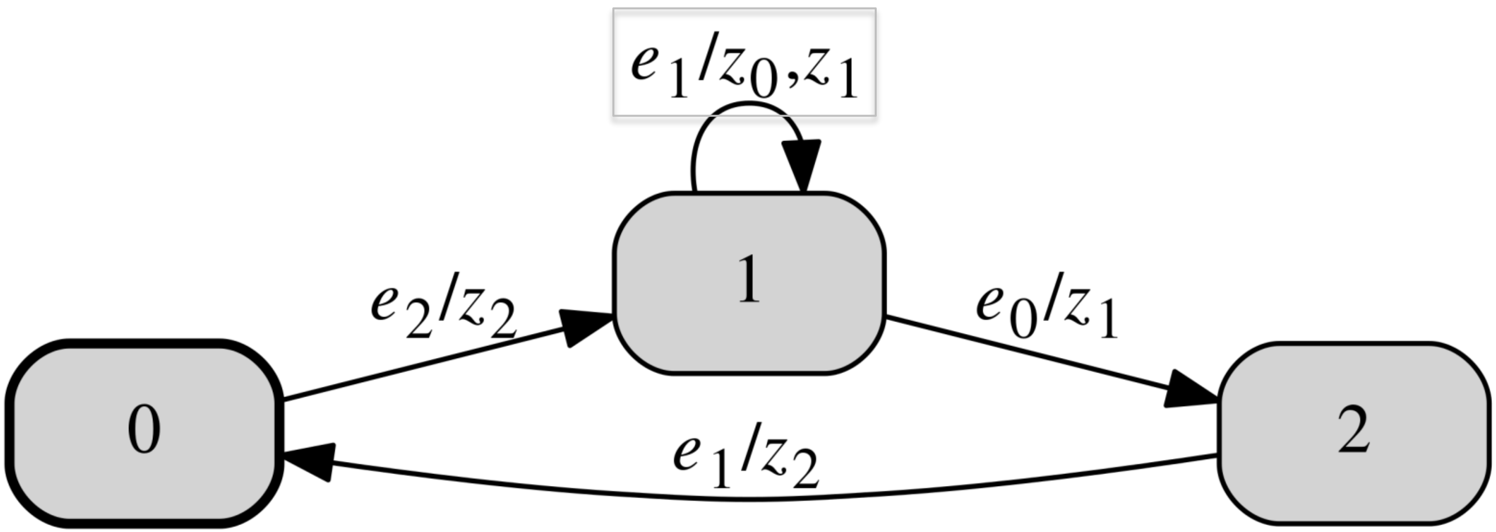
- Propositional variables: elementary statements
- Boolean logic operators: ∨, ∧, ¬, →
- Temporal operators
  - $X(f)$ – f has to hold in the next state
  - $F(f)$ – f has to hold in some state in the future
  - $G(f)$ – f has to hold for all states
  - $U(f, g)$ – f has to hold until g holds
  - …

Specification inference



Only very simple properties

Finite-State Machines

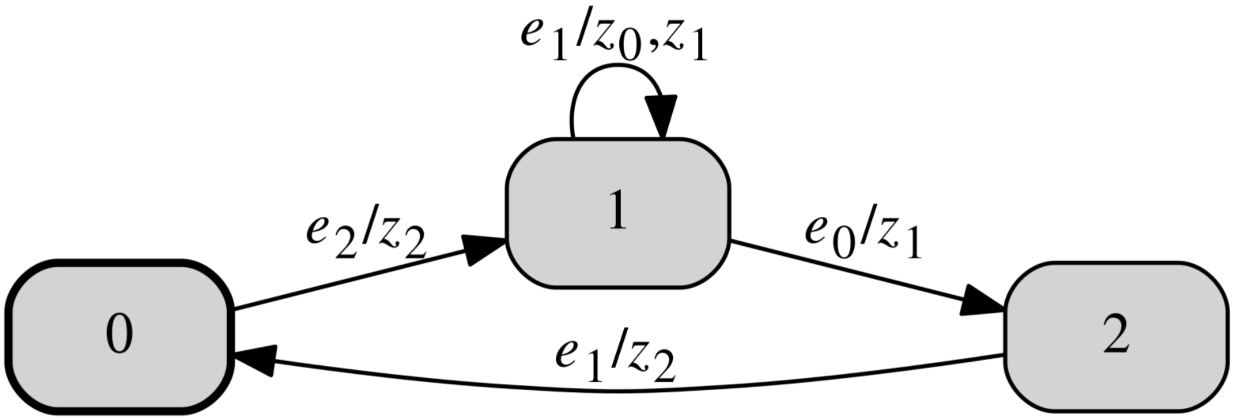


event → e_1 / z_0, z_1 ← output actions

LTL for FSMs

Propositional variables

- wasEvent(*e*) for all events *e*
- wasAction(*z*) for all output actions *z*



$G\ (\text{wasEvent}(e_2) \rightarrow \text{wasAction}(z_2))$

Problem statement

Find some non-trivial “interesting” LTL properties (formulas) of a given FSM

- All formulas **must** hold for input FSM
- Short formulas are better than long ones
- Should not hold for FSMs similar to the input FSM

Proposed approach

- Use Genetic Programming (GP)
- Evolve a population of LTL formulas
- **Express constraints using several fitness functions**
- Multiobjective optimization

ITMO UNIVERSITY

Main challenge

- Design a set of fitness functions that result in proper LTL properties

FF #1: Formula must hold for input FSM

- Main search objective
- Use model checker to check formula ***f*** against FSM ***a***

$$F_1(f) = r(a, f) = \frac{\text{number of verified transitions}}{\text{number of transitions}} \in [0,1]$$

FF #2: Minimal formula weight

- Measure structural complexity of a formula
- Operators $O = \{\vee, \wedge, \neg, \rightarrow, X, F, U, R\}$
- Propositional variables

S = {wasEvent(e) for all $e \in E$} U {wasAction(z) for all $z \in Z$}

FF #2: Minimal formula weight (continued)

- Each operator and variable are assigned weight W
- $W(s) = w_s$ for $s \in S$
- $W(o(\mathrm{arg}_1, [arg_2])) = w_o + W(arg_1)\ [+W(arg_2)]$

$$F_2(f) = \frac{1}{W(f)} \in [0,1]$$

ITMO UNIVERSITY

FF #3: Random FSMs

- Idea: if a large number of randomly generated FSMs satisfy an LTL formula, it is meaningless
- Generate a number of random FSMs with the same interface as the input FSM $a_1, \ldots, a_{N\text{sample}}$

$$F_3(f) = \frac{1}{1 + \sum_{1}^{N_{\text{sample}}} r(a_i, f)^2}$$

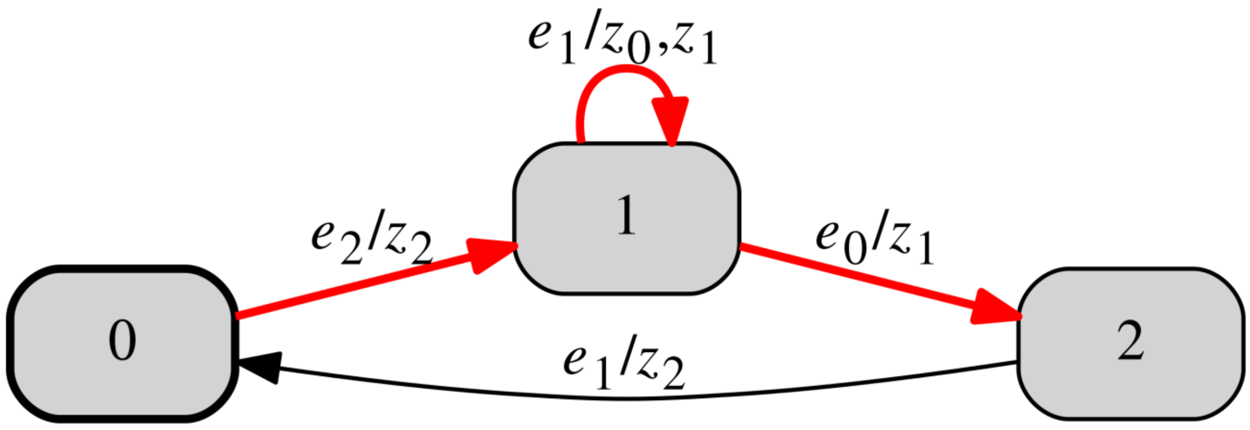
FF #4: Mutants of input FSM

- Idea: if a formula is not violated by a small change in the FSM, it is not so “interesting”
- Generate random mutants of the input FSM $m_1, \ldots, m_{N\text{sample}}$
- Mutation operators
  - Change transition end state
  - Add/delete transitions

$$F_4(f) = \frac{1}{1 + \sum_{1}^{N_{\text{sample}}} r(m_i, f)^2}$$

FF #5: FSM constructed from scenarios

- A scenario is a finite path in an FSM



- Example: $\langle e_2, (z_2)\rangle; \langle e_2, (z_0, z_1)\rangle; \langle e_0, (z_1)\rangle$

FF #5: FSM constructed from scenarios (continued)

- Derive random scenarios of fixed length from input FSM ***a***
- Use fast exact algorithm to construct an FSM ***a**** from scenarios
- Note: ***a**** probably differs from ***a***
- Note: not all formulas that are true for ***a*** are true for ***a****

$$F_5(f) = 1 - r(a^*, f)$$

FF #6: Mutants of FSM constructed from scenarios

- Same as FF #4, but mutants are generated from the FSM constructed from scenarios

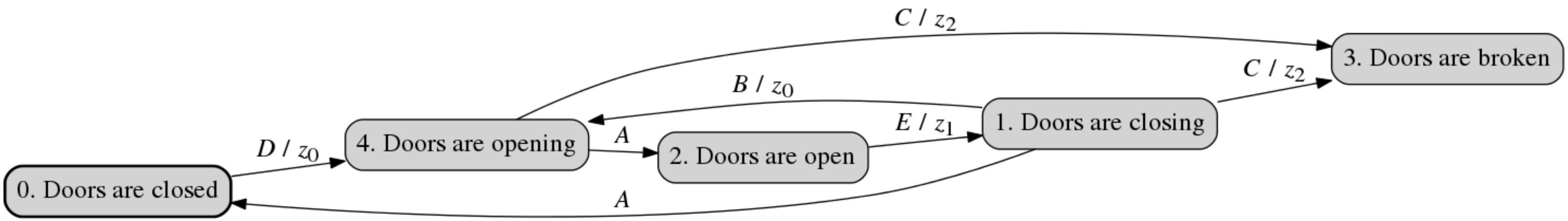
Implementation

- ECJ library used for EA implementation
- Multiobjective EAs: NSGA-II and SPEA2
- Standard GP operators

https://cs.gmu.edu/~eclab/projects/ecj/

Experiments

- Case study: Elevator doors control FSM
- Input events: A, B, C ,D, E
- Output actions: z_1, z_2, z_3
- 17 manually created LTL formulas



Original LTL properties

$$G(\text{wasEvent}(D) \rightarrow \text{wasAction}(z_0))$$
$$G(\text{wasEvent}(E) \leftrightarrow \text{wasAction}(z_1))$$
$$G(\text{wasEvent}(C) \leftrightarrow \text{wasAction}(z_2))$$
$$G(\text{wasEvent}(B) \rightarrow \text{wasAction}(z_0))$$
$$G(\text{wasEvent}(A) \rightarrow X(\text{wasEvent}(D) \vee \text{wasEvent}(E)))$$
$$G(\text{wasEvent}(D) \rightarrow X(\text{wasEvent}(A) \vee \text{wasEvent}(C)))$$
$$G(\text{wasAction}(z_0) \rightarrow X(\text{wasEvent}(A) \vee \text{wasEvent}(C)))$$

Experiments goal

- Goal: infer formulas similar to manually created ones
- But how do we measure the quality of inferred formulas?
- Introduced two empirical metrics
  - Coverage metric
  - Mutants metric

Coverage metric

- $\{f_{old}\}$ – original manually created formulas
- $\{f_{new}\}$ – inferred formulas

1. Derive scenarios from original FSM ***a***
2. Model inference: build FSM ***a'*** from scenarios *and* **$\{f_{new}\}$**
3. Metric: how many formulas from **$\{f_{old}\}$** does **a'** satisfy?

$$c_{\text{cover}} = \frac{\sum_{f \in \{f_{\text{old}}\}} r(a', f)}{\left|\{f_{\text{old}}\}\right|}$$

Mutants metric

- $\{f_{old}\}$ – original manually created formulas
- $\{f_{new}\}$ – inferred formulas

1. Generate $M' \leq 1000$ different mutants of original FSM ***a***
2. Ratio of mutants that violate at least one formula from **$\{f_{old}\}$**

$$n_{unsat}^{old} = \frac{1}{M'} \sum_{1}^{M'} \left(1 - \min_{f \in \{f_{old}\}} \lfloor r(m_i, f) \rfloor \right)$$

3. **Metric:**

$$c_{mut} = \frac{n_{unsat}^{new}}{n_{unsat}^{old}}$$

Experimental setup

- Tried both NSGA-II and SPEA2
- EAs run for 50 generations
- Population size = 500
- Result of experiment: all formulas in Pareto front
- Each experiment repeated 20 times
- FF_1 and FF_2 in all experiments, all combinations of the rest

Experimental data

№	F_3	F_4	F_5	F_6	$100 \cdot c_{\text{cover}}$, %	$100 \cdot c_{\text{mut}}$, %	**Time, s.**
1	-	-	-	-	44.1 / 44.1	53.4 / 38.5	60 / 14
2	-	-	-	+	64.7 / 58.8	49.6 / 36.6	170 / 78
3	-	-	+	-	73.5 / 70.6	65.3 / 58.0	133 / 84
4	-	-	+	+	88.2 / 88.2	77.5 / 83.6	521 / 2493
5	-	+	-	-	58.8 / 58.8	55.3 / 49.2	152 / 159
6	-	+	-	+	73.5 / 79.4	71.0 / 74.0	889 / 2898
7	-	+	+	-	88.2 / 79.4	78.6 / 79.4	579 / 2197
8	-	+	+	+	88.2 / **88.2**	83.2 / **86.4**	1894 / 4618
9	+	-	-	-	53.0 / 61.8	42.4 / 42.0	64 / 17
10	+	-	-	+	67.6 / 64.7	44.7 / 46.6	158 / 108
11	+	-	+	-	88.2 / 82.4	71.4 / 69.5	141 / 211
12	+	-	+	+	88.2 / 88.2	77.5 / 80.9	632 / 2025
13	+	+	-	-	67.6 / 58.8	66.4 / 56.9	236 / 195
14	+	+	-	+	64.7 / 79.4	71.0 / 69.1	796 / 2259
15	+	+	+	-	**88.2** / 88.2	**87.8** / 85.5	876 / 1775
16	+	+	+	+	88.2 / 82.4	84.0 / 83.6	1618 / 4724

Experimental results

- NSGA-II and SPEA2 yield similar formula quality
- SPEA2 is much faster than NSGA-II
- Config #8 = {all but FF_3} is best for NSGA-II
- Config #15 = {all but FF_6} is best for SPEA2
- Significance validated using Wilcoxon signed-rank test

Varying other parameters

- Use SPEA2 with config #15
- Varied population size from 100 to 1000

Pop size	100	250	500	1000
$100 \cdot c_{learn}$, %	23	86	86	86
$100 \cdot c_{mut}$, %	13	79	96	96

- Change number of generations from 25 to 200
  - No significant changes

Larger example

- ATM control FSM
- 12 states
- 14 events
- 13 output actions
- 30 LTL formulas
- Mutants metric: $100 \cdot c_{mut} = 65\ \%$
- Coverage metric: infeasible

Results

- Proposed GP-based approach for inferring LTL properties of FSMs
- Feasibility demonstrated on two examples using two empirical quality metrics
- Approach is able to infer up to 100 % of human-written LTL formulas

Future work

- Couple with existing model inference algorithms

Acknowledgements

- This work was financially supported by the Government of Russian Federation, Grant 074-U01.

ITMO UNIVERSITY

Thank you for your attention!

Daniil Chivilikhin
Ilya Ivanov
Anatoly Shalyto
chivdan@rain.ifmo.ru