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

Executable program → Model inference algorithm → Program model

Test cases → Model inference algorithm

Finite-state machine

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: $\lor$, $\land$, $\neg$, $\rightarrow$
- 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

Executable program → Specification inference algorithm → Temporal properties

Test cases

Only very simple properties
Finite-State Machines

- Event \( e_1 / z_0, z_1 \)
- Output actions

Diagram:

- State 0
- Transition \( e_2 / z_2 \) to State 1
- Transition \( e_1 / z_2 \) to State 2
- Transition \( e_0 / z_1 \) from State 1

States:
- 0
- 1
- 2
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
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 $a$

$$F_1(f) = r(a, f) = \frac{\text{number of verified transition } s}{\text{number of transition } s} \in [0,1]$$
FF #2: Minimal formula weight

✔ Measure structural complexity of a formula
✔ Operators $O = \{\lor, \land, \neg, \rightarrow, X, F, U, R\}$
✔ Propositional variables

$S = \{\text{wasEvent}(e) \text{ for all } e \in E\} \cup \{\text{wasAction}(z) \text{ for all } z \in Z\}$
FF #2: Minimal formula weight (continued)

- Each operator and variable are assigned weight $W$
- $W(s) = w_s$ for $s \in S$
- $W(o(\text{arg}_1, [\text{arg}_2])) = w_o + W(\text{arg}_1) \ [+W(\text{arg}_2)]$

$$F_2 ( f ) = \frac{1}{W(f)} \in [0, 1]$$
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 $a_1, \ldots, a_{N_{\text{sample}}}$

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

\[
F_4(f) = \frac{1}{N_{\text{sample}}} \left[ 1 + \sum_{i=1}^{N_{\text{sample}}} r(m_i, f)^2 \right]
\]
A scenario is a finite path in an FSM

Example:

\[ \langle e_2, (z_2) \rangle; \langle e_2, (z_0, z_1) \rangle; \langle e_0, (z_1) \rangle \]
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

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

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

\[
C_{\text{cover}} = \frac{\sum_{f \in \{f_{\text{old}}\}} r(a', f)}{|\{f_{\text{old}}\}|}
\]

✓ $\{f_{\text{old}}\}$ – original manually created formulas
✓ $\{f_{\text{new}}\}$ – inferred formulas
Mutants metric

1. Generate $M' \leq 1000$ different mutants of original FSM $\alpha$

2. Ratio of mutants that violate at least one formula from $\{f_{\text{old}}\}$

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

3. Metric:

\[
    C_{\text{mut}} = \frac{n_{\text{new unsat}}}{n_{\text{old unsat}}}
\]
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\textsubscript{1} and FF\textsubscript{2} in all experiments, all combinations of the rest
### Experimental data

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

<table>
<thead>
<tr>
<th>Pop size</th>
<th>100</th>
<th>250</th>
<th>500</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 · $c_{\text{learn}}$, %</td>
<td>23</td>
<td>86</td>
<td>86</td>
<td>86</td>
</tr>
<tr>
<td>100 · $c_{\text{mut}}$, %</td>
<td>13</td>
<td>79</td>
<td>96</td>
<td>96</td>
</tr>
</tbody>
</table>

- 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: $100 \cdot c_{\text{mut}} = 65\%$
✔ 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.
Thank you for your attention!

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