A Method for Automatic Runtime Verification of Automata-Based Programs

Oleg Stepanov
oleg.stepanov@gmail.com

Anatoly Shalyto
shalyto@mail.ifmo.ru

Saint Petersburg State University of Information Technologies, Mechanics and Optics

Problem Statement

Design a method for runtime verification of automata programs

Existing Approaches

- Rely on static verification, most commonly on *model checking*
- Use available verifiers like *SPIN* and *Bogor*
- Build Kripke structure which has exponential size of the program used to build it

Existing Approaches: Performance

- SPIN-based method can perform verification of models containing 100 to 500 automata depending on the number of transitions
- Methods based on model checking take exponential time to verify a system of automata

Runtime Verification

Runtime verification is verification of program execution traces which can be run in parallel with the verified program

Advantages of Runtime Verification

– Allows for verification of larger systems of automata
– Verifies *implementation*, not model
– Can be used for soft handling of exceptional cases in critical applications

Method Performance

Depends on trace size

Depends on formula complexity

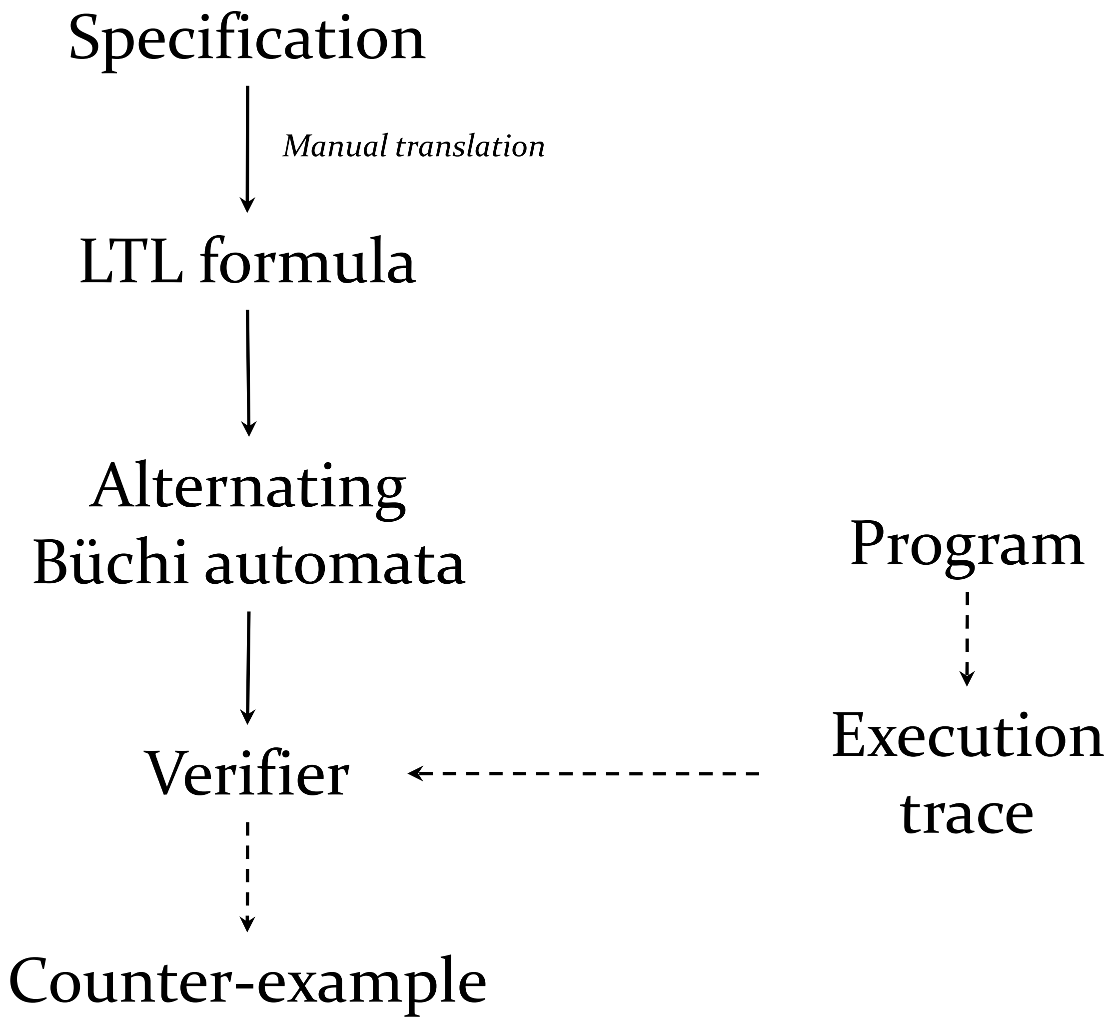
Does *not* depend on program complexity

Method Drawbacks

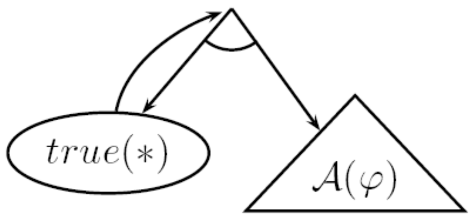
Does not guarantee valid program

Even worse for parallel programs

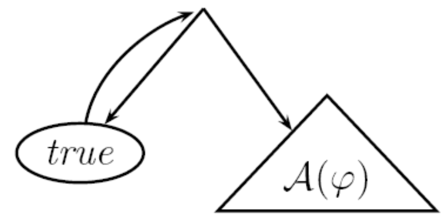
Runtime Verification: Workflow



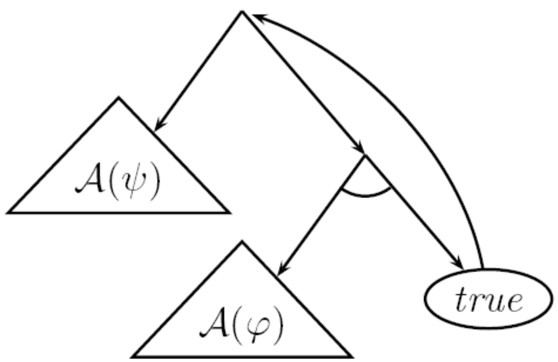
Samples of Alternating Büchi Automata



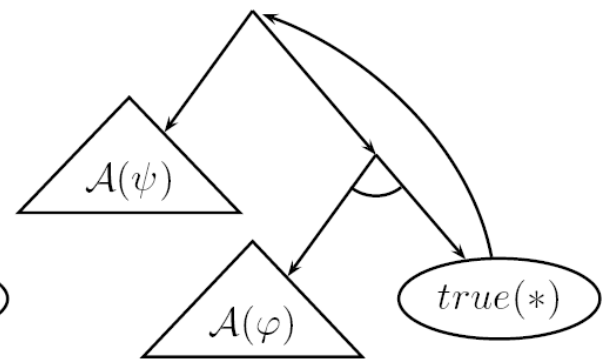
$\mathcal{A}(\Box\,\varphi)$



$\mathcal{A}(\Diamond\,\varphi)$



$\mathcal{A}(\varphi\,\mathcal{U}\,\psi)$



$\mathcal{A}(\varphi\mathcal{W}\psi)$

Traversal of Alternating Büchi Automata

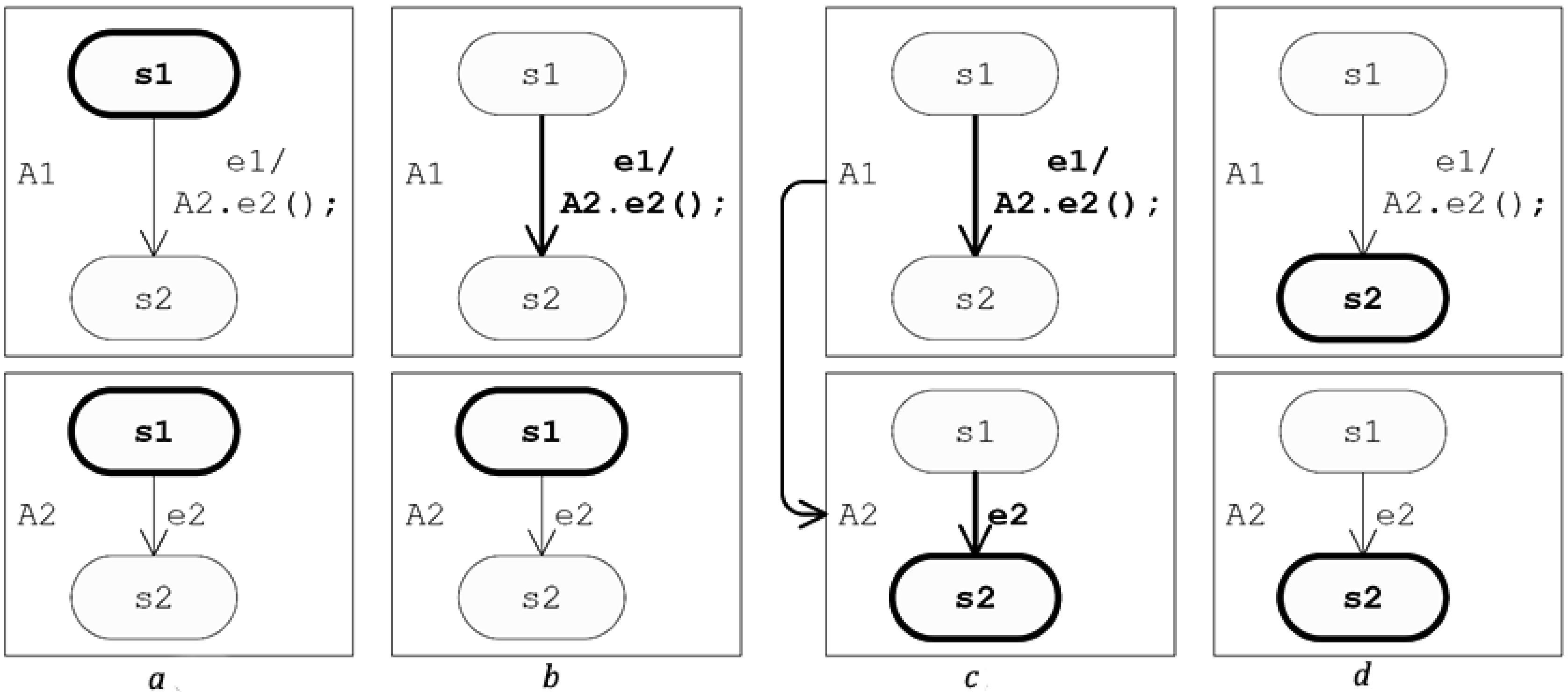
1. Depth-first traversal

   ↑ Choice depends on formula ↓

2. Breadth-first traversal
3. Reverse traversal

   Optimal choice when entire trace is available

Systems of Mealy Automata



Components of the Method

1. Trace construction algorithm
2. Set of atomic propositions
3. Algorithm for evaluating propositions at each trace point

Trace Construction

$$s_{1,1}s_{2,1}\left[e_{1,1}x_1\overline{x_2}s_{1,2}\right]\left[e_{1,2}\overline{x_1 x_2}\left[e_{2,3}x_1x_2z_1s_{2,2}\right]s_{1,3}\right]$$

Trace header *Handling section* *Nested section*

Root event *Variable values* *End state*

Atomic Propositions

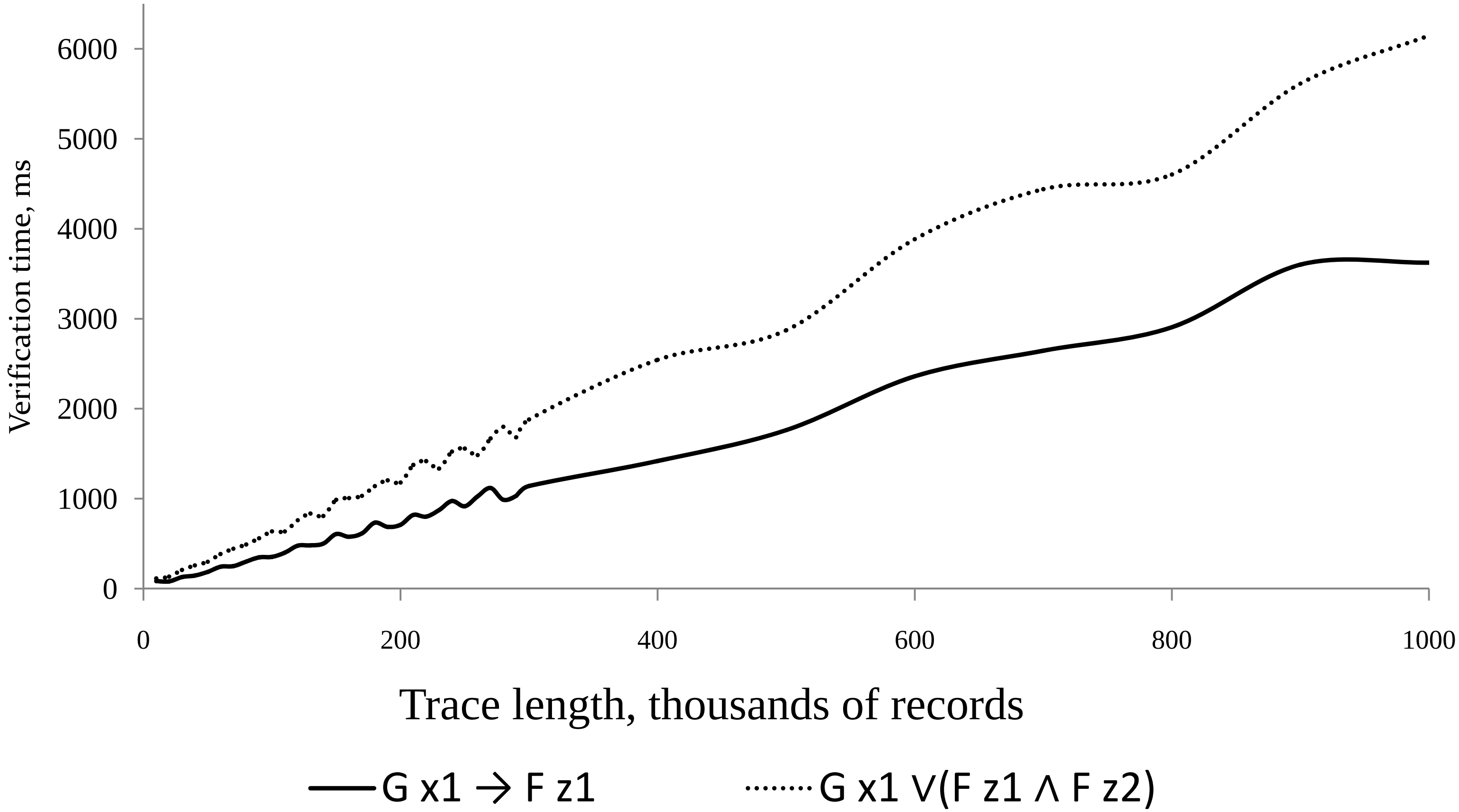
$e_{i,j}$ — *i-th* automaton is handling event e_j

x_i — value of input variable x_i is *true*

z_i — output action z_i is performed

$s_{i,j}$ — *i-th* automaton is in state $s_{i,j}$

Method Performance



Further Research

- Apply test input generation algorithms
- Build an efficient implementation for breadth-first algorithm
- Apply to real large systems
- Investigate applicability of automata programming in dynamically executed environments

Questions?

oleg.stepanov@gmail.com
shalyto@mail.ifmo.ru

http://is.ifmo.ru