The SIMPA tool
SIMPA Infers Models Pretty Accurately

A model inference toolbox

Roland GROZ, Karim HOSSEN
LIG, Université Grenoble Alpes, France

Outline

- SIMPA functions
- SIMPA architecture
  - packages, files
- TODO (by students): assignment

SIMPA offers

- Algorithms for inference of Mealy machines
  - L_M* (Angluin DFA adapted by Shahbaz to Mealy) table-based
  - Z-quotient (Petrenko, Li, Groz…): tree based (vs table-based)
  - NoReset Learner (Groz, Simao…), Combinatorial (Oriat), RS
- Algorithms for EFSM (data & NonDeterminism)
  - With Weka or built-in datamining algorithms
- Counterexample algorithms
  - Finding counterexamples (random walks…)
  - Processing counterexamples (e.g. suffixes 1 by 1…)
- Drivers to interface with real systems
  - HTTP, SIP, SAML…

SIMPA also offers

- Test programs
  - For performance evaluation of algos, collecting stats
- Random machine generators
  - Purely random
  - Combined with counters
- Transparent drivers
  - Glass-box automata

SIMPA’s features for Web security

- Crawlers
- Automatic generation of Web drivers
  - Based on crawling
  - Sets alphabet=actions on Web app
- Detection of XSS vulnerabilities
  - Reflected or stored values.

Outline

- SIMPA functions
- SIMPA architecture
  - packages, files
- TODO (by students): assignment

SIMPA data

- Written in Java, developed under Eclipse
  - Subclasses: Mealy, EFSM
- 241 Java files, 39000 loc
- No test ☹
- Hardly any doc
  - See lecture on testing & machine learning
  - Associated research papers

SIMPA: structure 1

- Automata
  - Subclasses: Mealy, EFSM
- Drivers: translating alphabet <-> pgm calls
  - Real drivers: to interact with real system (e.g over network)
  - Transparent drivers: automaton is in fact a SIMPA Java object that simulates a black box *(useful for testing)*

SIMPA: structure 2

- Learners: implement learning algorithms
  - Mealy: L_M* (= table), or Z-quotient (= tree)
  - noReset, combinatorial, RivestSchapire: 3 algorithms that do not reset the automaton
  - EFSM: 2 algos (table and tree)
- Main
  - SIMPA.java: main launcher
  - Other drivers (Test, Stats): deprecated

SIMPA: other packages

- Drivergen, Crawler:
  - Used for automatic generation of drivers for Web applications, prior to learning
- Auxiliary packages
  - Stats: to collect stats for performance eval.
  - Tools: various utilities, including loggers
  - Datamining algorithms: used for EFSM
  - Detection: used to find security vulnerabilities

Outline

- SIMPA functions
- SIMPA architecture
  - packages, files
- TODO (by students): assignment

Assignment

- Global goal: enhance project with tests and testing facilities to enable e.g. regression testing
  - Propose a test architecture (language framework, test harness, file structure etc)
  - Propose a test strategy
  - Implement it, add corresponding files (e.g. have /test at same level as /src)
  - Populate with tests

Restrictions

- Too many things to test, so concentrate on main targets
  1. Mealy algorithms
  2. Web EFSM if time permits
  3. (for the greedy ones): web crawling & inference
- A minima: system tests
  - If time permits, unit tests, integration tests…

Criteria for assessment

- Ease of use, integration into project
- Fault coverage: report bugs (e.g. with a bugtracker that you set up, such as Mantis, Bugzilla)
- Program (feature) coverage:
  - Algos covered
  - Options covered
  - Number & quality of tests

Deliverables

- Updated package with test
- Fault reports
- Documents
  - Test architecture, strategy and organization
  - Report on bugs and other problems found
- Bonuses
  - Documenting SIMPA itself (~ReadMe for newcomers)
  - Experience report on this “legacy” test project